INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

ESTG

DICOM WEB VIEWERS:
AN APPROACH AND PERFORMANCE ANALYSIS

VISUALIZADORES DICOM WEB:
UMA ABORDAGEM E UMA ANÁLISE DE DESEMPENHO

Masters Thesis

Hugo André Faria Pereira

Escola Superior de Tecnológica e Gestão

DICOM web viewers: An approach and performance analysis
Hugo André Faria Pereira
2024

Instituto Politécnico
de Viana do Castelo

Hugo André Faria Pereira

DICOM WEB VIEWERS:
AN APPROACH AND PERFORMANCE ANALYSIS

VISUALIZADORES DICOM WEB:
UMA ABORDAGEM E UMA ANÁLISE DE DESEMPENHO

Nome do curso de Mestrado

Mestrado em Engenharia Informática

Trabalho efetuado sob a supervisão de

Professor Doutor Pedro Miguel Teixeira Faria

Professor Doutor Luís Miguel Cabrita Romero

Maio de 2024

Resumo

Imagens médicas desempenham um papel crucial na saúde moderna, facilitando o diagnóstico e o tratamento de várias condições médicas. O padrão para a gestão de dados de imagens médicas é o protocolo DICOM (Digital Imaging and Communications in Medicine). Os visualizadores web de DICOM oferecem plataformas flexíveis e acessíveis para que os utilizadores possam visualizar e analisar imagens DICOM remotamente.

Com as constantes melhorias e inovaçoes nas técnicas de imagens medicas a necessidade de visualizadores DICOM eficientes e versáteis torna-se ainda mais exigente. Assim, esta dissertação realiza um estudo abrangente sobre visualizadores DICOM, incluindo o seu processo de desenvolvimento, uma comparação de desempenho e a avaliação das funcionalidades disponíveis, testando-os em diferentes computadores e browsers.

O estudo investiga o processo de desenvolvimento do Cornerstone3D, uma framework de visualização de DICOM files baseada em web com capacidades avançadas de renderização 2D e 3D. A arquitetura, as funcionalidades e o desempenho da framework são analisadas, fornecendo insights sobre o processo de desenvolvimento de um visualizador web DICOM.

O estudo inclui uma pesquisa abrangente de visualizadores DICOM web existentes, avaliando as suas capacidades de renderização 2D e 3D, compatibilidade com navegadores e desempenho em várias configurações de hardware. OHIF, PostDICOM, MEDDream, e outros visualizadores DICOM populares são avaliados, destacando os seus pontos fortes, fracos e adequação para diferentes casos de uso.

Além disto, a dissertação realiza uma comparação detalhada de desempenho entre vários visualizadores DICOM, elucidando a sua velocidade e eficiência. Esta análise comparativa pode servir como um guia para profissionais de saúde, investigadores e

desenvolvedores na escolha do visualizador DICOM mais adequado às suas necessidades específicas.

A dissertação contribui para o avanço da tecnologia de visualizadores DICOM ao oferecer uma análise abrangente das soluções existentes e propor uma abordagem para a visualização 2D e 3D de imagens médicas. As descobertas fornecem insights valiosos para profissionais de saúde, investigadores e desenvolvedores que procuram aproveitar visualizadores DICOM para uso pessoal ou profissional.

Palavras-chave: DICOM. Cornerstone3D. Visualizador Web. Desempenho.

Abstract

Medical imaging plays a crucial role in modern healthcare, facilitating the diagnosis and treatment of various medical conditions. The standard for managing image data in healthcare is the DICOM (Digital Imaging and Communications in Medicine) protocol. DICOM web-viewers provide flexible and accessible platforms for users to view and analyze DICOM images remotely.

As medical imaging techniques continue to improve, the need for efficient and versatile DICOM (Digital Imaging and Communications in Medicine) viewers becomes even more demanding. Thus this dissertation conducts a comprehensive study of DICOM visualizers, including their development process, a performance comparison, and available features testing them across different computers and browsers.

The study delves into the development and evaluation of Cornerstone3D, a web-based DICOM viewer framework with advanced 2D and 3D rendering capabilities. The framework's architecture, features, and performance are analyzed, providing insights into the development process of a DICOM web viewer.

The study includes a comprehensive survey of existing DICOM viewers, evaluating their 2D and 3D rendering capabilities, browser compatibility, and performance on various hardware configurations. OHIF, PostDICOM, MEDDream, and other popular DICOM viewers are assessed, highlighting their strengths, weaknesses, and suitability for different use cases.

Furthermore, the dissertation conducts a detailed performance comparison among various DICOM viewers, elucidating their speed, and efficiency. This comparative analysis can serve as a guiding beacon for healthcare professionals, researchers, and developers seeking to select the most suitable DICOM viewer tailored to their specific requirements.

The dissertation contributes to the advancement of DICOM viewer technology by offering a comprehensive analysis of existing solutions and proposing an approach for 2D and 3D visualization in medical imaging. The findings provide valuable insights for healthcare professionals, researchers, and developers seeking to leverage DICOM viewers for personal or professional use.

Keywords: DICOM. Cornertsone3D. Web-Viewer. Performance.

Acknowledgements

I would like to express my deepest gratitude to everyone who supported me throughout the development and completion of this dissertation.

First and foremost, I would like to thank Professor Pedro Faria and Professor Luís Romero for their invaluable guidance, patience, and encouragement. Their expertise and feedback were instrumental in shaping this work. Furthermore, I would also like to thank these two professors for the chance to work on the HOUDINI project, as a research fellow, which allowed me to introduce myself to scientific research and the publishing of my first scientific article, published at the 2023'IWSSIP conference. They also helped me with my second published article entitled "Medical Imagery Processing, Reconstruction and Visualization: A Web Platform Approach" as a result of this dissertation, published at the CISTI'2024 conference. Additionally, I am grateful to them for allowing me to submit a third article to the Journal of Digital Imaging - Springer, which is currently under revision.

I am also grateful to my colleagues and fellow students who offered their assistance, shared their knowledge, and provided a supportive and collaborative environment. Your camaraderie made this journey more enriching and enjoyable.

My heartfelt thanks go to my friends, whose constant support and encouragement have been a source of strength throughout this process. Your belief in me and your understanding during the challenging times have meant the world to me.

Finally, I would like to extend my deepest appreciation to my family. To my father and my mother, your unwavering love, encouragement, and sacrifices have made all of this possible. Your constant support has been the foundation upon which I have built my achievements.

Contents

List of Figures

List of Tables

List of Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
API	Application Programming Interface
CPU	Central Processing Unit
CSS	Cascading Style Sheets
CT	Computed Tomography
DICOM	Digital Imaging and Communications in Medicine
DWV	DICOM Web Viewer
FPS	Frames Per Second
FViewer	Fast DICOM Viewer
GPU	Graphics Processing Unit
HTML	Hypertext Markup Language
JS	JavaScript
JSON	JavaScript Object Notation
MRI	Magnetic Resonance Imaging
OHIF	Open Health Imaging Foundation
PACS	Picture Archiving and Communication System
RAM	Random Access Memory
UI	User Interface

Chapter 1

Introduction

The process of diagnosing a patient by medical professionals has evolved significantly over the years, progressing from a diagnosis reliant solely on a doctor's intuition to the utilization of comprehensive body scans [1]. It is only natural that the analysis of these scans would transition from a 2D to a 3D state through computer software. By uploading our 2D scans and having them processed and reconstructed into a 3D state, their analysis is streamlined, enabling a thorough examination of the entire body rather than relying on intuition.

With continuous innovations and technological advancements, the quality and efficiency of patient diagnosis have markedly improved. Nevertheless, in the present era, medical professionals extract crucial information by reviewing medical images in a 2D format. A 3D counterpart would empower them to navigate through a person's body scan more effectively.

In the rapidly evolving landscape of modern medicine, the proficient management and interpretation of medical images are crucial for patient care. Digital Imaging and Communications in Medicine, commonly known as DICOM [2], has emerged as the most popular language of medical imaging [3], transforming the way healthcare professionals access and analyze diagnostic images. DICOM was developed to address interoperability challenges in medical imaging, ensuring the seamless sharing of images and associated patient data across diverse medical equipment, including X-ray machines, MRI (Magnetic Resonance Imaging) scanners, CT (Computed Tomography) scanners, and healthcare information systems [2]. This standardized format enables healthcare

professionals to consistently access, view, and analyze medical images, irrespective of the equipment's manufacturer or origin.

DICOM viewers serve as the bridge between cutting-edge medical imaging technology and the imperative for seamless data interoperability across various healthcare devices and systems. These versatile software applications are crafted to unleash the full potential of DICOM, allowing healthcare providers to access, manipulate, and analyze medical images with precision and ease. In doing so, they facilitate collaborative decision-making, expedite diagnoses, and enhance patient outcomes.

In addition to traditional DICOM viewers, there is a burgeoning demand for DICOM web viewers [4] with a study revealing that 78% of patients feel more involved in their healthcare when they can access their medical data [4]. These web-based applications offer healthcare professionals the flexibility to access and analyze medical images from any location with an internet connection. DICOM web viewers have become particularly invaluable in the era of telemedicine and remote healthcare services. They enable healthcare providers to securely access and share medical images with colleagues and specialists worldwide in real-time. With the ability to view and manipulate DICOM images through a web browser, healthcare professionals can collaborate effectively, make swifter diagnoses, and provide timely interventions, ultimately elevating the standard of patient care.

1.1 Context

As mentioned in the preceding section, there have been concerted efforts to develop software capable of processing medical images and reconstructing them into 3D visualizations through DICOM viewers. Identisoft, an enterprise specializing in information and communication technology solutions for the health sector, leverages its expertise in medical imaging data management to drive innovation. One notable solution is an application designed to visualize 2D medical images in the DICOM (Digital Imaging and Communications in Medicine) format, a standard pivotal in facilitating the seamless exchange and interoperability of medical images and associated information.

In collaboration with the Escola Superior de Tecnologia e Gestao de Viana Do Castelo

(ESTG-IPVC) institution, both parties proposed the development of a web application for medical image visualization that integrates both 2D and 3D capabilities. In this context, it was decided to plan and develop a web application that may incorporate advanced tools, including volume rendering and 3D reconstruction, and leverage the Cornerstone3D [5] library—a lightweight JavaScript library specifically crafted for visualizing medical images in modern web browsers, with purported good performance.

1.2 Problem Statement and Motivation

Initially working with "Identisoft" a project was proposed to display 2D Medical Images in a 3D format using the new technology "Cornerstone3D" to develop a website capable of the previous. Eventually, as more applications were found capable of the defined objective and issues with the still-under-development tech, a gathering of available DICOM viewing tech and a comparison was in order. Other comparisons have been made but none compare the performance of the viewers on various browsers and computers only the features available on each viewer.

1.3 Objectives

Two sets of objectives were set for this dissertation. Working with Identisoft several requirements/goals were defined to be achieved from the development of a web platform capable of viewing DICOM files. They are as follows:

- Research and explore other solutions in the area.
- Work with DICOM P10 anonymized images (from CT studies) provided by Identisoft.
- Stream the slices of a volume and view them in real-time as they are being loaded.
- View the same volume in different orientations such as axial, sagittal, and coronal without having to reload, the entire volume again (minimum memory footprint);
- View oblique slices in a volume.

- Render different blends of the same volume (e.g., MIP (maximum intensity projection) and average intensity projection).
- Implement a Crosshairs tool.
- Fuse and overlay multiple images such as PET/CT fusion.
- Test the developed platform with end-users.

A second set of objectives was set for the evaluation of the available DICOM web options on the market. They are as follows:

- Research and explore other web DICOM viewers and their technology.
- Research and explore previous surveys, comparisons, and evaluations.
- Define a number of metrics to evaluate DICOM viewer software/technology's.
- Gather web viewers and create a comparison table containing various features that would be important to the end-user.
- Test each software's performance using the same dataset of DICOM images using multiple browsers and machines.
- Create comparison tables of the performance of each viewer.
- Derive conclusions by evaluating the array of features offered by each viewer and conduct performance analysis encompassing the computers, browsers, and DICOM viewer software.

1.4 Organization

This section provides an overview of each chapter in the dissertation.

- Introduction: This chapter introduces the topic of the dissertation, outlines its objectives and scope, and discusses the research methodology employed.
- Concepts, Languages, Frameworks, and Tools: This chapter discusses the theoretical concepts, programming languages, frameworks, and tools used in the development and evaluation of the DICOM viewers.

- Literature Review: This chapter reviews existing literature and research relevant to DICOM viewers, exploring their features, performance, and usability.
- Image Reconstruction and Volume Rendering: This chapter delves into the principles and techniques of image reconstruction and volume rendering using the Cornerstone technology, as well as the development of a DICOM viewer using the technology.
- DICOM Viewer Analysis: This chapter presents an analysis of various DICOM viewers, including their performance and features.
- Conclusions: This chapter summarizes the key findings of the dissertation, discusses their implications, and suggests directions for future research in the field.

1.5 Methodology

Numerous research techniques are currently in use, and each is suitable and useful for a particular field of study. The topic and the research questions being examined determine the best strategy to employ [6]. The research method used is the Design Science Research (DSR) method which had its origins in engineering, and other applied sciences. This research method is a creative and innovative problem-solving activity whose main product is the creation of new technologies [7]. It has been called the "improvement research method" to underline the fact that this method focuses on problem-solving or improving solution performance [8].

This methodology has 6 main steps:

- Problem identification and motivation - The problem and the motivation were presented in section 1.1
- Definition of objectives - The objectives were defined in section 1.3 and the state of the art in chapter 3
- Design and development of the artifact - The design and development of the software can be seen in section 6 and the metrics and performance comparisons in section 6
- Demonstration - The demonstration of the software can be seen in section 6 and the tables generated from the viewers are in section 6

- Assessment/Evaluation - Results driven from the development of the software are in chapter 6 and the conclusions from the survey and performance tests in chapter 6

Chapter 2

Concepts, Languages, Frameworks and Tools

This chapter introduces and explains some of the concepts, languages, frameworks, and tools that are talked about/used throughout the document. Understanding the previous is essential to comprehend what advantages and disadvantages each includes. The selection process for the tools can be found in Chapter 2.3.

2.1 Concepts

2.1.1 DICOM

Digital Imaging and Communications in Medicine (DICOM) is a standard utilized in the field of medical informatics, which has evolved into the most extensively implemented and supported communication standard for medical imaging [9]. It ensures interoperability between different devices and systems, allowing its users to access and share patient images and information seamlessly across various platforms and institutions.

2.1.2 PACS

Picture Archiving and Communication System is referred to as PACS, it is a medical imaging technique that offers easy access to and affordable storage of pictures from several modalities, including CT, MRI, and X-rays [10]. PACS systems can receive DICOM

images, store them in their database, and make them accessible to authorized users such as radiologists, physicians, and other healthcare professionals.

The physical and scheduling constraints associated with conventional film-based image retrieval, distribution, and display are eliminated with the adoption of PACS, which enables the rapid and simple retrieval of images and other relevant data. The process can be seen in Figure 1.

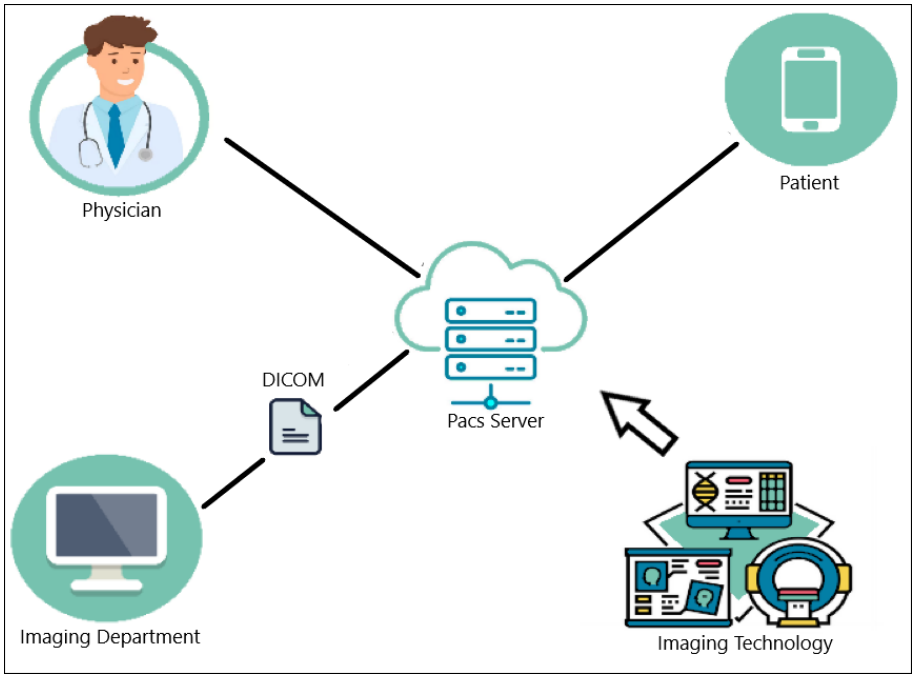


Figure 1: PACS Server Architecture and distribution process.

2.1.3 Frontend & Backend

Frontend and backend development are two critical components of web development. Frontend deals with what users see and interact with directly, including design and functionality using languages such as HTML, CSS, and JavaScript while the backend handles the behind-the-scenes functionality such as processing requests and data, and ensuring smooth communication between the server and the client.

2.2 Languages & Frameworks

2.2.1 Javascript

JavaScript is a versatile and powerful programming language primarily used for creating dynamic and interactive web content, used on most websites and available on all modern browsers. It was initially developed by Brendan Eich in 1995 and has since become one of the fundamental building blocks of the modern web, being the most omnipresent language in history [11].

2.2.2 Typescript

TypeScript is a superset of JavaScript that adds static typing and other advanced features to the language. Developed by Microsoft, TypeScript aims to enhance JavaScript development by providing optional static typing, which helps catch errors early in the development process and makes code more robust and maintainable [12]. It is used extensively in Cornerstone3D.

2.2.3 Cornerstone.js

Cornerstonejs a JavaScript library that provides a framework for building web-based medical imaging applications. It offers tools for displaying, interacting with, and manipulating medical images such as X-rays, MRIs, and CT scans within a web browser [13]. Cornerstone.js is designed to be highly customizable and extensible.

2.2.4 Cornerstone3D

Cornerstone3D is an extension of Cornerstone.js specifically focused on supporting 3D medical imaging data. Leveraging Cornerstone3D and its associated libraries, such as Cornerstone3DTools, enables the accomplishment of a broad spectrum of imaging tasks [5].

2.3 Tools

The selection/screening process for the tools involved thorough exploration/searches on popular search engines, GitHub, and a review of viewers used on previous surveys. Through diligent research and evaluation, a large number of viewers were explored, from these, only viewers with online capability were considered, either through a direct online demo or installable software. 16 web-viewers passed the initial screening. Some of the chosen tools have standalone versions; however, this survey will only analyze their web-based parts. Any available standalone versions will be mentioned in the survey table. To be included in the survey, each viewer must pass a screening process. The viewer must either be free or, have a free trial available for usage, not be a duplicate entry, and have a working online demo or be able to run on a local server. The selection process is illustrated in Figure 2.

2.3.1 Open Health Imaging Foundation

The Open Health Imaging Foundation, commonly referred to as OHIF, is a prominent organization dedicated to advancing healthcare through innovative medical imaging technologies [5], their viewer is a versatile and open-source medical imaging viewer designed to facilitate the visualization and analysis of medical images. They have recently released OHIF v3 which now uses the also newly released Cornerstone3D which builds on top of Cornerstonejs (a web-based medical imaging platform) to include 3D viewing capabilities.

2.3.2 Med3Web

Med3Web is a high-performance web application for sophisticated medical volumetric data display (in 2D and 3D modes) [14]. The most recent version is compatible with mobile browsers (Android Chrome) and desktop browsers that support WebGL. It uses DICOMParser from Cornerstonejs, Daikon, a JavaScript DICOM reader, Xtk, a framework/toolkit for scientific and medical visualization that makes use of WebGL to provide sophisticated 3D medical visualization, and Three.js, a JavaScript framework for WebGL visualization.

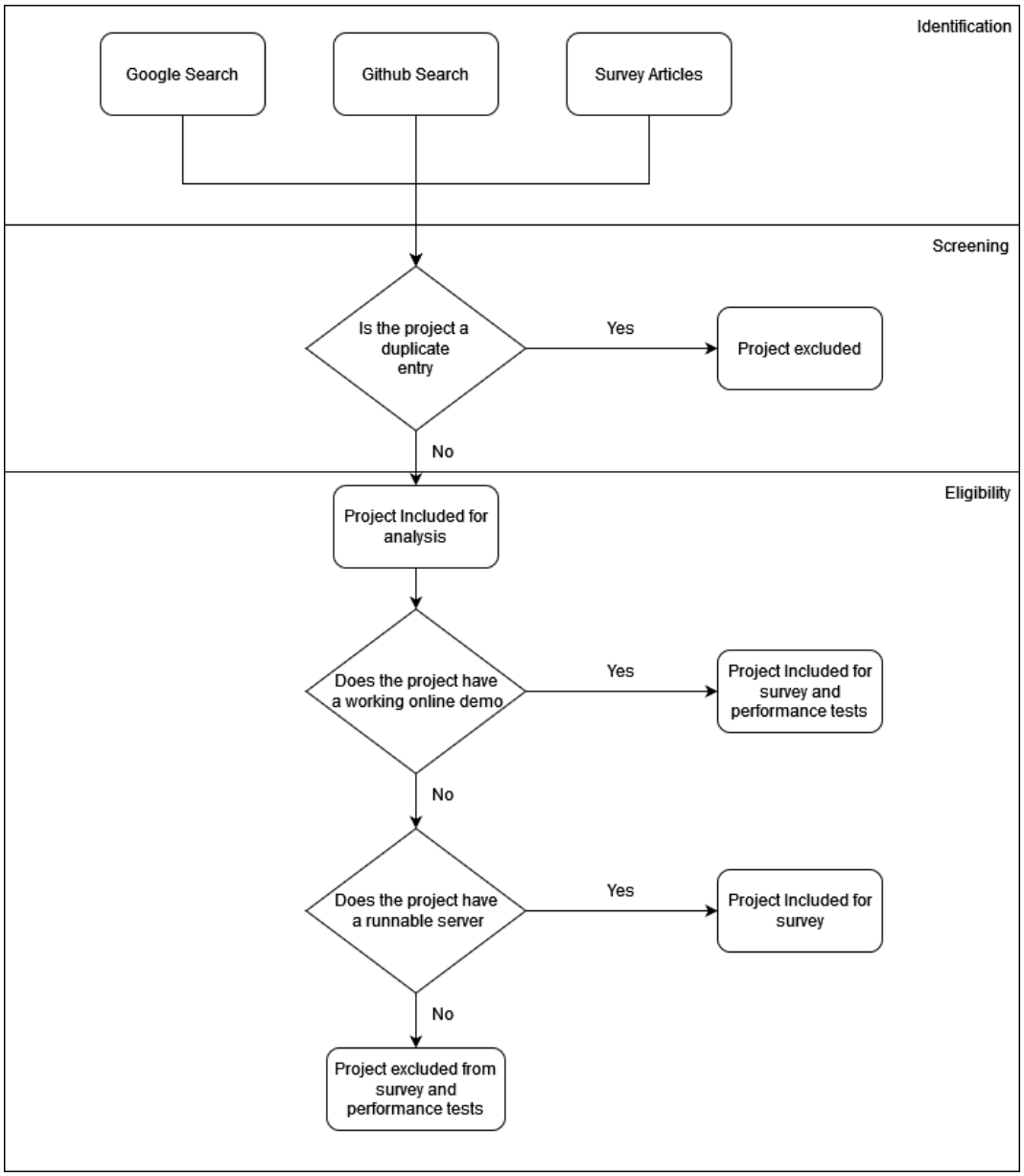


Figure 2: Tool selection process.

2.3.3 Papaya online DICOM viewer

Papaya is a medical research picture viewer that runs only on JavaScript and Daikon for DICOM support [15]. Supported data types for this orthogonal viewer include overlays, atlases, DTI (Diffusion Tensor Imaging), and VTK(Visualization Toolkit) surface data. With numerous display, menu, and control options, the Papaya UI is highly adaptable. It can be used locally as a shared file or on a web server.

2.3.4 DICOM Web Viewer

The DWV (DICOM Web Viewer) is a free and open-source medical image viewer with a minimal footprint [16]. It solely employs the use of JavaScript and HTML5 allowing it to operate on multiple platforms.

2.3.5 Weasis

Weasis is a highly modular standalone and web-based DICOM viewer with a multifunctional architecture [17]. It is a widely used clinical viewer in healthcare, with hospitals, health networks, multi-center research trials, and patients as their targeted consumers. It is a cross-platform open-source software and through the OpenCV library, it enables high-quality renderings with high performance.

2.3.6 IMAIOS DICOM Viewer

IDV - IMAIOS DICOM Viewer is a medical image viewer app for web browsers and Android devices [18]. It accepts all DICOM files, such as ultrasound, scanner, MRI, PET, etc. The app includes numerous tools for manipulating and measuring graphics, such as scrolling and image alteration.

2.3.7 PostDICOM

PostDICOM Viewer is a medical imaging software application that allows the examination and handling of DICOM files [19]. It provides a user-friendly platform for healthcare practitioners and radiologists to access, evaluate, and analyze DICOM pictures, with capabilities such as multi-planar reconstruction, measuring tools, and the

ability to annotate and store patient data. It makes medical picture interpretation easier, making it a crucial tool for medical diagnosis and patient care. It also provides its own PACS server in the form of cloud storage allowing for any photos obtained from patients to be directly uploaded to the server and then accessed from anywhere with the appropriate credentials.

2.3.8 VoxelX

VoxelX was a medical imaging platform and healthcare ecosystem that uses blockchain and artificial intelligence (AI) to enhance how doctors, researchers, and patients engage with medical data. This technology was intended to make medical images and related data more accessible and safe by streamlining storage, sharing, and analysis [20]. VoxelX seeked to improve medical imaging interpretation, diagnostic accuracy, and the interchange of medical information by combining 3D visualization, deep learning algorithms, and blockchain technology. The project however is no longer accessible and no new news has been given by the development team since 2018, leading to its exclusion from the survey.

2.3.9 FViewer

FViewer is a Free, online cloud file viewer that takes no download or installation and supports 12 file formats one of which is the DICOM format [21].

2.3.10 Open DICOM Viewer

Open DICOM Viewer (ODV) is a lightweight multi-platform program for viewing DICOM images. It is portable as it may be used immediately from removable media, and supports web view display for displaying DICOM images on web pages using Java technologies [22]. This Viewer was built using Java. It was tried on two Windows 11 machines and results indicate that it no longer works on the newer Windows version. On Windows 10 the application opens but it was not capable of loading DICOM 3.0 files.

2.3.11 Osimis

The Osimis Basic Web Viewer extended Orthanc an open-source DICOM server with a Web viewer of medical images [23], with more advanced features. It has however been acquired by "Deepc" and is no longer available for free nor does the new version feature a free trial version available and was excluded from the survey.

2.3.12 DICOMBinator

DICOMBinator was created over 24 hours for the SXSW Interactive Hackathon [24]. "Through an easy-to-use interface, this web-based tool enables users to examine and annotate DICOM medical pictures" The app was built using javascript allowing "instantaneous dissemination of comments and visual annotations to other users" enabling then instantaneous communication. This application however was built with node 0.6 and this version requires 32-bit, an attempt was made to run the web app using node 0.8.28 but without success, the app also has a demo on its GitHub page however it no longer works, leading to its exclusion from the survey.

2.3.13 Hida DICOM Viewer

Hida is a JavaScript DICOM viewer and analysis tool. It was developed as part of a research internship at the Academic Medical Center in Amsterdam [25]. Its goal was to provide a future-proof rebuild of a Liver Uptake measurement algorithm [25] but grew as a general-purpose web-based DICOM tool. No demo was provided and the viewer failed to install on two different computers, leading to its exclusion from the survey.

2.3.14 Slice:Drop Online DICOM Viewer

Slice:Drop (Sdrop) is a web-based application that allows for the instant viewing of scientific and medical imaging data in 3D [26]. It supports a variety of file formats including DICOM. It was programmed with JavaScript using WebGL for the rendering.

2.3.15 Oviyam

Oviyam is a web-based open-source DICOM Viewer [27], it is pre-packaged for deployment with JBoss. It was built using the dcm4che toolkit and script.aculo.us framework and programmed in the Java Language.

2.3.16 DicomLibrary and MedDream

DicomLibrary is a free online medical DICOM image or video file-sharing service for educational and scientific purposes [28]. All DICOM files are anonymized before they are uploaded into the cloud, no personal information is ever revealed ensuring the user's or patient's privacy. It uses the MedDream DICOM viewer, an "HTML 5 zero-footprint DICOM VIEWER" aimed at making diagnoses, viewing, archiving and transmitting medical images [28, 29]. Together these tools can be used to upload and view your own DICOM files.

2.4 Conclusion

This chapter allows for an understanding of the concepts, languages, frameworks, and tools used throughout the rest of the project. Languages such as JavaScript and TypeScript, alongside frameworks like Cornerstone.js and Cornerstone3D, empower developers to visualize and manipulate medical images seamlessly within web browsers. The surveyed applications were presented, ranging from versatile open-source viewers like OHIF to specialized applications like Med3Web and Papaya, each tailored to address medical image visualization and analysis needs. The next chapter will go over the state of the art of DICOM and Medical Image Viewing.

Chapter 3

Literature review

In this chapter, we present a history of DICOM and the changes it suffered over the years followed by a history of viewers made to view DICOM files and finally other evaluations/surveys of the available viewers for consumers.

3.1 DICOM History

The National Electrical Manufacturers Association (NEMA) established a joint committee in 1983 with the goal of creating a standard for connecting displays and related equipment to various manufacturers' medical imaging equipment[9]. ACR/NEMA Standard Version 1.0, the initial version, was released in 1985. The original version 1.0 of the standard was followed by two updates, released in October 1986 (No. 1) and January 1988 (No. 2)[30]. Versions 1.0 and 2.0 both worked with point-to-point connections, which posed a challenge for communication networks that do not employ wholly dedicated channels. A revision of the standard ensued, leading to the emergence of DICOM Version 3.0 in 1993 [9, 3, 31]. This revised iteration revolutionized imaging system communications by transitioning from a point-to-point connection environment to a networked one. DICOM Version 3.0 facilitates cost-effective connectivity across expansive geographic areas, leveraging existing network infrastructure. In 2004 Picture Archiving and Communication System (PACS) originated for usage in healthcare facilities with the goal of managing and storing the data generated within radiology departments [32]. PACS established a standardized method for obtaining, storing, and

sending medical pictures and adopted the DICOM standard which specifies the handling, storing, and transmission of medical imaging data. The DICOM Standard has undergone numerous revisions since its 1993 publication and its history is publicly available [31] as shown in Table 1. from the years 1995 to 2024.

These updates/changes are done mainly through working groups. Some of the groups with importance to radiology include cardiac and vascular information and ultrasound [3]. When the changes are approved, they become an official part of the standard. Having gone through these changes has made DICOM the most widely implemented and supported communications standard for medical imaging [3].

3.2 DICOM Viewer frameworks

Numerous software solutions over the years dealt with the DICOM format for displaying medical images, including ITK (Insight Toolkit)[33] and VTK (Visualization Toolkit)[34]. ITK, developed by the US National Library of Medicine in 1999, is an open-source software toolkit for image processing and analysis in C++. It often collaborates with VTK, an open-source system for 3D computer graphics and visualization. An example of this collaboration can be seen in the DICOM viewer MITK (Medical Imaging Interaction Toolkit) which integrates algorithms from ITK with visualizations from VTK, creating tailored applications for medical image analysis [33, 35]. ITK has two types: single DICOM reading and sequential DICOM. ITK uses the itkGDCMImageIO class to read slices and the itkGDCMSeriesFileNames class to create lists of filenames and determine which slices of a common volumetric dataset belong to the same series it provides a large amount of medical image processing and analyses algorithms but it does not provide algorithms for the purpose of data visualization [33]. While VTK is an open source, freely available software system for 3D computer graphics, modeling, image processing, volume rendering, scientific visualization, and information visualization, it also encapsulates a related DICOM reading class: vtkDICOMImageReader [34]. These two are some of the most popular libraries which are often used together to create DICOM viewing software [33].

Web-based DICOM viewers have been making substantial progress through the years

Table 1: DICOM Standards Evolution (1995-2024).

Year	Description
1995	Protocols for nuclear medicine, X-ray angiography, and ultrasound were introduced to DICOM to support cardiology imaging demands, and the DICOM Standards Committee was reorganized to represent all medical specialties that use imaging.
1996	With the addition of the Modality Worklist service by DICOM, workflow management in the imaging department was standardized.
1997	Radiation therapy information objects were added.
1999	Endoscopy and dermatology were included in the standard, and it was made possible for image annotations to be presented consistently across display systems.
2000	Further standardization on structured data, analytic results, and clinical observations, and addition of internet security mechanisms through secure communication profiles.
2001	Mammography CAD (Computer Aided Detection) added, and new security improvements.
2003	Multi-frame enhanced image formats added, with the objective to support future generations of MR and CT imaging techniques. Support for DVD (Digital Versatile Disc) media added.
2004	WADO (Web-access to DICOM objects) added, dentistry and ophthalmology joined DICOM, support for USB and flash memory media.
2005	Radiation Dose Structured Reports (RDSR) added for x-ray based imaging.
2006	Deformable spatial registration added.
2007	Radiation Dose for CT (RDSR) added.
2008	Breast tomosyndissertation ("3-D mammography") added.
2009	3D ultrasound added.
2010	Whole slide imaging and surgical planning information objects added.
2011	Bluray media exchange added.
2013	Second-generation RESTful web services created to retrieve, store, and query DICOM images and rebranding of web services to DICOMweb™.
2014	Radiopharmaceutical Radiation Dose Reporting (RRDSR) added.
2015	Server-based Rendering added to WADO-RS, enabling web clients to make requests for the presentation of DICOM images and video. Tractography Results storage added. Imaging report templates using the HL7 Clinical Document Architecture (CDA) added. Wide-Field Ophthalmic Photography images, brachytherapy Delivery Instruction objects, and support for AVC/H.264 (MPEG-4) video added.
2016	CT Protocol Storage, small Animal Acquisition Context headers, and HEVC/H.265 (MPEG4) video support added. Adult Echo Measurement SR objects updated/simplified.
2017	Volumetric and Blending Presentation States, Patient Dose Estimate reports (P-RDSR) based on RDSR data for individual patients and Ophthalmic OCT angiography (OCT-A) imaging added. NCI AIM to DICOM SR transcoding is defined to facilitate the management of image markup.
2018	3D Manufacturing - STL encapsulation added image-based medical 3D-printing workflows. Contrast Injection SR and Multi-Energy CT Image Storage added TLS ciphersuites updated to match security recommendations.
2019	DICOMweb documentation re-organized. DICOMweb Thumbnails service and Realtime Video Streaming (DICOM-RTV) added.
2020	Dermoscopy, Neurophysiology Waveform, and 2nd Gen. Other non-C-Arm RT supplements added to the DICOM Standard. Further support for Virtual Reality, Augmented Reality, and Mixed Reality was added.
2021	Whole Slide Imaging Annotation, and cone-beam CT (CBCT) acquisition support added
2022	TLS security was updated, and the DICOM Conformance Statement was revised
2023	Support added for Confocal Microscopy, Photoacoustic Imaging, 32-bit ECG Waveform, and JPEG 2000 (HTJ2K)

as a result of rising demand, usage, and potential for browser-based software. Solutions such as Med3Web [14], or VTK.js challenge the web part of the existing viewers mostly using WebGL on instances per viewport and because of GPU memory limitations, they do not scale well for scenarios like PET/CT hanging protocols, which could need for more than ten viewports to be displayed at once [5]. Cornerstone's rendering engine was built from the ground up for efficient GPU memory usage as an attempt to have greater performance than their other online counterparts. Cornerstone3D does not deal with loading images; instead, it delegates this work to image and volume loaders registered with Cornerstone3D such as CornerstoneWADOImageLoader [5].

Besides browser viewers, there is also plenty of standalone software for viewing and processing DICOM based images such as MITK [35] a combination of the ITK and VTK software which was released in 2016 and is now on its newest version v.2023.04 it was created with the goal to drastically cut down on the time and effort needed to construct specifically tailored/custom, interactive applications for medical image analysis [36] and the case presented by H. Dong [37] who similarly to the previous, encompasses both VTK and ITK tool-kits to used to create a medical image reconstruction platform targeting DICOM type files [37]. The article delves deeper into the explanation of the tool-kits, describing both VTK and ITK in detail. It concludes that "ITK can easily perform medical image processing and VTK can conveniently display the outcome of the reconstruction" [37]. However, due to the lack of acceleration algorithms usage, the program itself will take a long time to process the entire reconstruction. Additionally, the article concludes that as the number of DICOM files increases, so does the time required to process the files. Thus, to reduce the reconstruction time, the quantity of DICOM slices must be reduced.

3.3 3D Visualization Techniques and Methods

In 2016, J. Tan presented an article [38] introducing key techniques in the medical image 3D visualization field and designed a system capable of visualizing 3D medical images, based on VTK. The article goes into detail on the Marching Cubes and Ray casting algorithms. The Marching Cubes algorithm, also known as the "Isosurface Extraction algorithm," is used to render isosurfaces in volumetric data [38]. The basic notion in the

algorithm consists of defining a voxel (which represents a value on a regular grid in three-dimensional space; essentially, it's a 3D cube used to create 3D models) as the sequence of the pixel values at the eight corners of a cube. If one or more pixels of a cube have values less than the user-specified isovalue, and one or more have values greater than this value, one knows the voxel must contribute some component of the isosurface. It is possible to generate triangular patches that divide the cube into regions inside the isosurface and regions outside by figuring out which edges of the cube the isosurface intersects. A surface representation is obtained by connecting the patches from all cubes on the isosurface boundary, as illustrated in Figure 3.

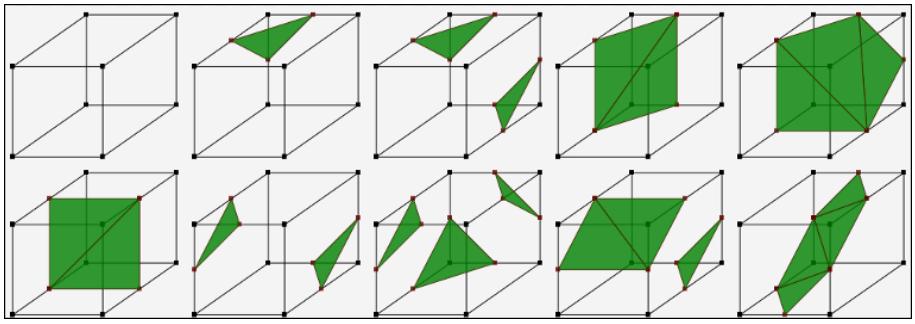


Figure 3: Marching cubes algorithm surface generation.

The vtkMarchingCubes function in VTK is a filter that, according to the VTK documentation, takes as input a volume (such as a 3D structured point set) and outputs one or more isosurfaces.

The Ray Casting Algorithm, on the other hand, uses ray-surface intersection to determine whether a point is inside a polygon. If the algorithm is applied to a polygon, it will trace a ray in any direction, counting the number of times the ray intersects with the polygon. If it intersects once, it's inside the polygon; if it never intersects (0), then it misses the polygon. The system created in this article's making integrates the Marching Cubes and Ray Casting visualization algorithms; it implements file reading, 3D reconstruction, and display of DICOM format of medical images using the VTK platform [38].

A paper by MH. Tran in 2016 [39] reviews the existing methods for building a 3D object from 2D medical image slices in DICOM format and afterward constructed an application based on the VTK library for building, visualizing, and handling a 3D object

from CT scanned slices, which allows interaction with the reconstructed object. Similar to the "Design of 3D Visualization System Based on VTK Utilizing Marching Cubes and Ray Casting Algorithm" [38], it explains the marching cubes algorithm and ray casting algorithm, reaching a comparison in the conclusion: "Each method has its own advantages and disadvantages." MC (Marching Cubes) offers some benefits, including the ability to easily make modifications, the ability to extract necessary surfaces, the capacity to render quickly, and the lack of memory restriction. The ray casting algorithm offers a straightforward implementation, flexible expansion, and high-quality images. Additionally, this method will access the entirety of the video memory, and after creating a 3D model, it considerably improves the image quality. However, the method's processing time requires lengthy computation. The article goes over another algorithm called "The Texture-Based Algorithm," which is another direct volume rendering method. This is the fundamental method in computer graphics to map an image onto a geometric object. Essentially, it takes an image and applies it over an object becoming its skin. By dividing a model into the XY, YZ, and XZ slices, this algorithm can be applied over each slice and, at the end, generates a 3D model after blending the color and opacity; background lighting; fine-tuning, and adjusting the model [39]. In comparison with the previous two algorithms, this algorithm runs the fastest and produces images with excellent sustainability. However, it is limited in texture memory making it less likely to be used when dealing with a large number of medical data slices. [39].

3.4 DICOM Viewer surveys

There have been many studies in the past that evaluate many of the older DICOM viewers, such as "Evaluation of commercial PC-based DICOM image viewer" which was published by Honea et al. in 1998 [40]. The study evaluated essential features like various windowing levels, measures of distance, and angles in the best free software that was available at the time. A study in 2003 by Horii made an analysis of viewing capabilities using supported DICOM object types, image processing techniques, and picture exporting capabilities [41]. Liao et al. carried out the first systematic evaluation in 2008 through which 21 software projects containing a GUI (Graphical User Interface) were examined

in terms of " support, portability, workability, usability, data import, data export, header viewing, two-dimensional, and three-dimensional imagine viewing" [42]. Most categories were divided by their "yes/no" capabilities, with a few exceptions which were graded based on percentage value. The previously mentioned surveys are rather old and outdated, and have had newer surveys based on their work such as "A Survey of DICOM Viewer Software to Integrate Clinical Research and Medical Imaging" [43] in 2015 which performs a comprehensive evaluation of state-of-the-art DICOM viewer software for medical usage. They defined use cases for DICOM with a certain level of abstraction, these use cases being split into: Central viewing where data is gathered from PACS and two-dimensional (2D) functionality is what's needed rather than sophisticated three-dimensional (3D) rendering; Decentral viewing where data is shared between sites [44] being a good fit for DICOM web viewers; Advanced viewing where in some circumstances, additional viewing features such as sophisticated 3D picture rendering and volumetric data processing is critical. Another survey in 2015 was made focusing on the loading of patient-specific 3D models [45], the survey uses a mix of qualitative methods such as subjective percentage values, and the Yes/No evaluation method. However, it's important to note that even this survey is 6 years old, and new tech has been developed and released since its writing. The most recent research found is from 2020. The first is focused on free DICOM viewers for use in veterinary medicine, carried out by Andreas Brühschwein [46], and has a different approach to the previous ones, as a group of observers was chosen and asked to perform some "selected tasks" with the various software and then score the operability of each tool using a scale of 1 to 10. This study also found that gender, computer skills, veterinary degree, and experience did not affect the grading of the viewers significantly. The second study is an "Evaluation of free, open-source, web-based DICOM viewers for the Indian national telemedicine service "[47] and goes over 6 viewers with web capabilities and available online documentation and demo. It is the first study to mention an old version of OHIF although this version did not have any 3D capabilities.

3.5 Conclusion

DICOM has continually adapted to meet the evolving needs of medical imaging, expanding its scope to encompass various modalities and functionalities. Various techniques for creating a 3D object from 2D ones exist each with different advantages and disadvantages. From the continuous demand and evolution, new solutions for the viewing of DICOM files are developed continuously. Numerous surveys are done updating on the previous with newer software. Although there are many surveys focused on the features of the viewers, an actual performance comparison between the multiple software solutions is lacking. The next chapter will go over developing a web app using Cornerstone3D.

Chapter 4

Image reconstruction and volume rendering

This chapter covers specific research and development on the DICOM web viewer application. It includes an exploration into volume rendering and image processing, the mechanisms by which Cornerstone handles these processes, a comparison between the older Cornerstone.js library, the more recent Cornerstone3D, and a standalone viewer (MITK). Additionally, this chapter outlines the proposed design for the website, the architecture created, and the developed prototype that allows for the importing and viewing of DICOM files using Cornerstone3D.

4.1 Planning and requirements gathering

The Cornerstone website has tutorials and guides on how to use the technology for developers to follow, some important characteristics of the technology are mentioned in these tutorials such as requiring to use outside aid to fill some of the areas that the technology does not such as loading images.

Table 2. is a comparison table between three DICOM viewers namely Cornerstone.js, Cornerstone3D and MITK, based on previous works [35][43][45], was created using parameters defined in prior research [43][45]. MITK and Cornerstone were among the DICOM viewing software covered in these studies, with Cornerstone3D being a recent addition. The rationale for choosing MITK and Cornerstone over other options lies in

MITK's extension of VTK and ITK, and Cornerstone's role as the precursor to Cornerstone3D, providing insight into the advancements introduced by the newer software.

The parameters selected for the table include:

- Web-based: Web applications accessible via modern browsers on client systems.
- Metadata: Viewing DICOM object metadata, including image resolution, study identifiers, and vendor-specific DICOM tags.
- Multi-platform: Software developed for use with multiple operating systems.
- Query-Retrieve: DICOM query and retrieve (Q/R) for actively requesting and gathering data from other DICOM nodes.
- Documentation: Availability of written software documentation, such as manuals.
- Wiki: Presence of a wiki web page for user reference.
- Forum: Web-based forum for user support.
- Measurements: Drawing geometric figures (e.g., lines) and analyzing them (e.g., distances, angles) with calibration information from DICOM headers.
- Annotations: Storing results of image viewing for later use through measurements or text annotations.
- Secondary Reconstruction: Functionality for reconstructing a secondary axis based on the primary direction for better visualization of specific structures.
- Slice Cube Volume Slices: Showcasing volumes effectively by independently moving various slice axes (e.g., transversal, sagittal, coronal) with displayed slices in a separate window.
- Volume Rendering: Direct representation of 3D image data with the ability to scale, translate, and rotate the volume.
- Surface Generation: Calculating surfaces of voxels with the same grey values using algorithms like marching cubes to enhance the visibility of image structures.

Table 2: DICOM viewing and processing software comparison.

	MITK	**Cornerstone**	**Cornerstone3D**
Interface			
Web	-	+	+
Multi-Platform	-	+	+
DICOM Capable	+	+	+
Support			
Documentation	+	-	+
Wiki	+	-	-
Forums	-	+	+
2D Viewing			
Metadata	+	-	-
Measurements	+	+	+
Annotations	+	-	-
3D Viewing			
Secondary Reconstruction	+	-	+
Slice Cube Volume slices	+	-	+
Volume Rendering	+	-	+
Surface Generation	+	-	+

From analyzing Table 2, it is possible to conclude that Cornerstone3D improves on the already existing library, mainly on the 3D Viewing area. These improvements can be found in detail in the next section. It is also important to note that while Cornerstone3D is still lacking in the 2D Viewing area, it delegates some of its work to Image Loaders and that while the library isn't capable of "seeing" DICOM metadata, there are separate libraries that allow for this process, see Fig. 4.

4.1.1 Image Loaders

Medical images are typically followed by a large amount of non-pixel-wise metadata, such as the pixel spacing of the image, the patient ID, or the scan acquisition date. This data is stored in the file header for some file types (e.g. DICOM) and can be read, parsed, and passed throughout the application. Others (like JPEG and PNG) require that this information be provided separately from the actual pixel data. However, since this can significantly improve performance, it is typical for application developers to

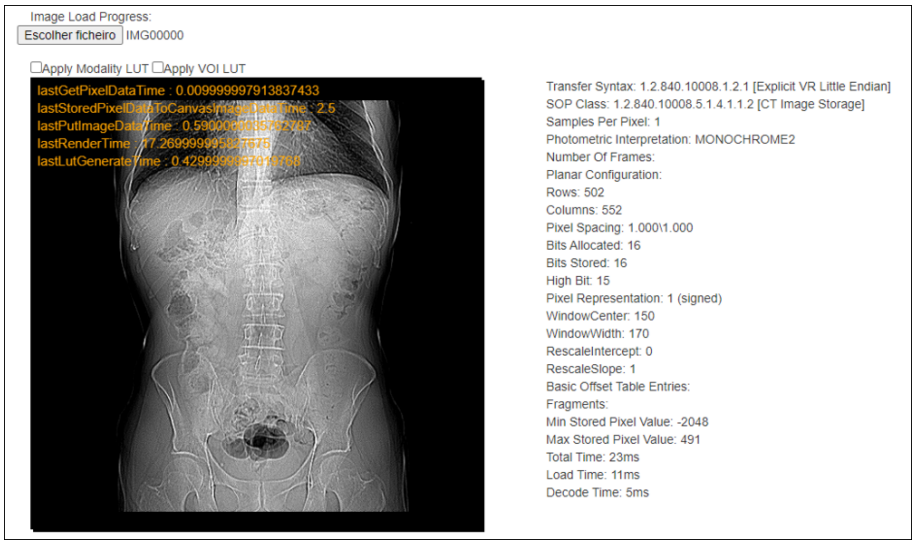


Figure 4: CornerstoneWadoImageLoader (DICOM Loader Example) for a DICOM file upload, provided by Identisoft enterprise.

provide metadata independently from the transmission of pixel data from the server to the client even for DICOM images (CornerstoneJS, 2023; Cruz, R., 2022). Cornerstone requires that Image Loaders return an Object containing a Promise (An object returned by an asynchronous function, representing the current state of the operation) which Cornerstone will use to receive the Image Object asynchronously, or an error if one has occurred. CornerstoneWADOImageLoader is one such loader, it loads DICOM images from a WADO-compliant server (Ziegler, E. et al., 2020). It can be installed and initialized using the code provided by Cornerstone in their documentation and tutorials. The final result can be seen in Figure 1, with the 2D representation on the left and the image metadata on the right. Internally, CornerstoneWADOImageLoader registers its wado-rs and Wado-uri image loaders to Cornerstone3D and uses DICOMParser to parse the metadata and pixel data. The CornerstoneWadoImageLoader provides a few examples for the uploading of DICOM images that were used with the DICOM images provided by the Identisoft enterprise, for testing purposes. This image loader, however, has been recently deprecated and Cornerstone3D has moved on to "dicomImageLoader" which, according to its backlog on "13/02/2024" still does not work with the examples provided by the Cornerstone team. However, in order to continue developing the work described, the CornerstoneWADOImageLoader continued to be used.

4.1.2 Heading Volume and Volume loaders

A volume is a 3D data array that has a physical size and orientation in space. It can be built by composing pixel data and metadata of a 3D imaging series or it can be defined from scratch by the application. A volume loader takes a volumeId (the volume identification) and other necessary data, much like image loaders do, and returns a Promise that resolves into a Volume. This Volume may be created from a single 3D array object or from a collection of 2D images (such as imageIds) (such as NIFTI format) [13], [48]. Cornerstone offers a volume loader that enables individual image slices to be streamed in, to fill an entire volumetric array (e.g. a portion of an MRI or CT acquisition). Essentially, reconstructing a 3D model from 2D images, as visualized in Figure 2, the user can still adjust the volume, while this slice-by-slice streaming is taking place. This loader was created/designed to accept imageIds and load them into a Volume.

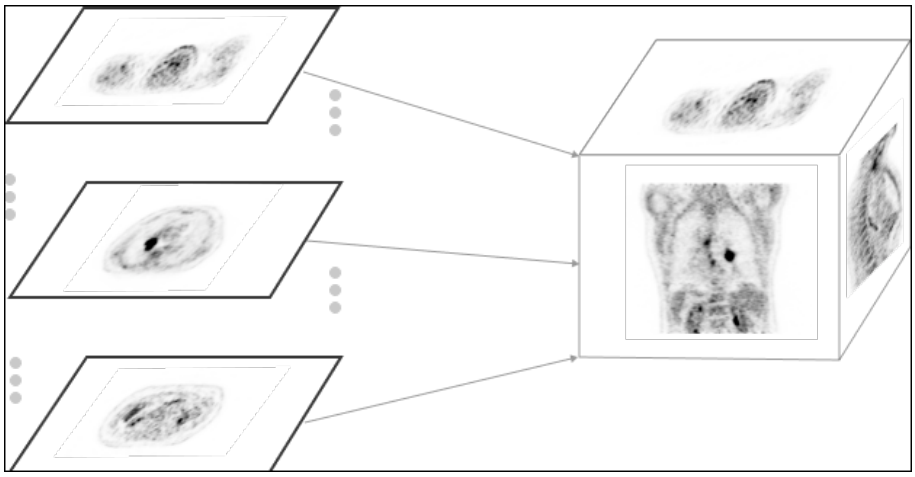


Figure 5: Volume Building - Visual reconstruction from 2D to 3D, picture available in Cornerstone web platform (CornerstoneJS, 2023).

Since a 3D Volume is comprised of 2D images (in StreamingImageVolume), the metadata for the volume is derived from the 2D image metadata. Therefore, this loader needs a first call to fetch image metadata. This allows us to render a volume as the 2D images are loaded, in addition to pre-allocating and caching a volume in memory (progressive loading). It is possible to avoid having to create an Image object for each imageId, by pre-fetching the metadata from all images (imageIds). Instead, insert the image's pixelData into the volume in the appropriate place. This ensures quickness and memory effectiveness (but comes at minimal cost of pre-fetching the metadata). Suppose

the objective is to return from the 3D format to 2D. In that case, a volume can implement functions to transform its 3D pixel data to 2D photos, without having to re-request them over the network, thanks to the StreamingImageVolume, which loads a volume based on a series of obtained images (2D). For instance, StreamingImageVolume takes an imageId and its imageId index and uses convertToCornerstoneImage to return a Cornerstone Image object (ImageId Index is required since the goal is to locate the imageId pixelData in the 3D array and copy it over the Cornerstone Image).

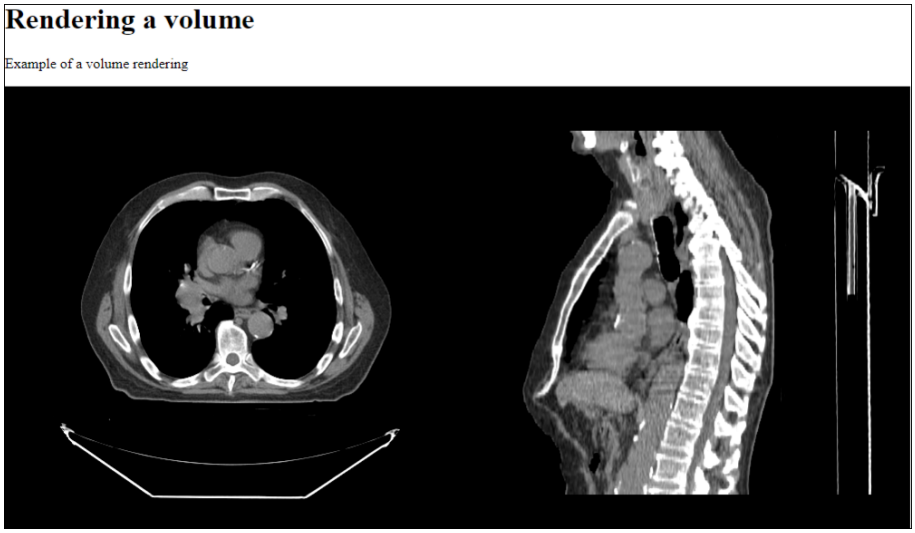


Figure 6: Volume Rendering - Result from the tutorial provided by Cornerstone for rendering a volume [13].

Once the 3D render is complete Cornertsone3D's rendering engine allows for the swapping of the different renders of the same volume seamlessly without any need for a reload, the rotation of of the rendered object also has no delays as illustrated in Figure 7.

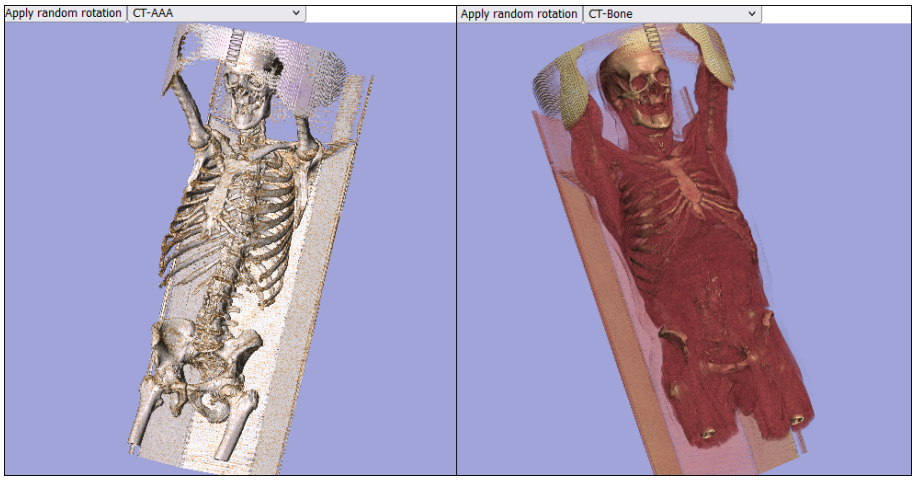


Figure 7: Swapping rendering mode and rotating the rendered object

4.2 Design phase

A Web platform prototype was created which contains a landing page where a regular user would initially start, then be sent to a projects page where the user would import their DICOM files into the web platform, which then allows for the viewing of each DICOM file loaded even during their import process. A login system would not be set up and the files and project would be lost upon ending the session, by closing the browser, thus ensuring the file's privacy and increasing ease of use, by not requiring a user account. In Fig. 8, the mockup is illustrated specifically showcasing the 2D reconstruction of DICOM files previously imported within the web platform. Users have a range of options for viewing them, including conversion into a 3D state, which unlocks additional unique 3D viewing options and functionalities. Additionally, the web platform has a documentation/help page to assist users in navigating the various viewing modes and understanding the website's functionality. A contact page is also provided, having information about the developers, in case users have unanswered questions after consulting the help tab.

The other pages created in the mockup can be found on B.1 on the Figures 19, 20, 21, and 22.

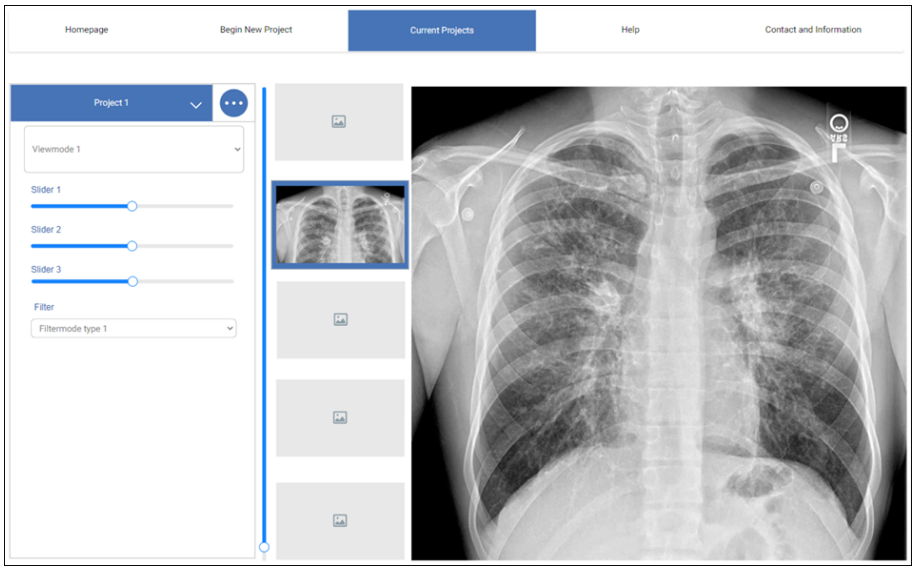


Figure 8: Prototype/mockup developed for the website.

4.3 Development phase

4.3.1 Architecture

The Web platform has the architecture shown in Figure 9. Initially, DICOM images are uploaded to the MIPVR platform, which uses an Image loader to load them, namely CornerstoneWadoImageLoader, then these files will be processed and rendered into an interactive 2D visualization, with an option to continue into a 3D visualization. Cornersonte3D powers the entire rendering process, and the website is built with HTML (hypertext markup language), JavaScript, and CSS (cascading style sheets) are the planned languages that will be used to support the website development, along with React technology, to make the process more agile.

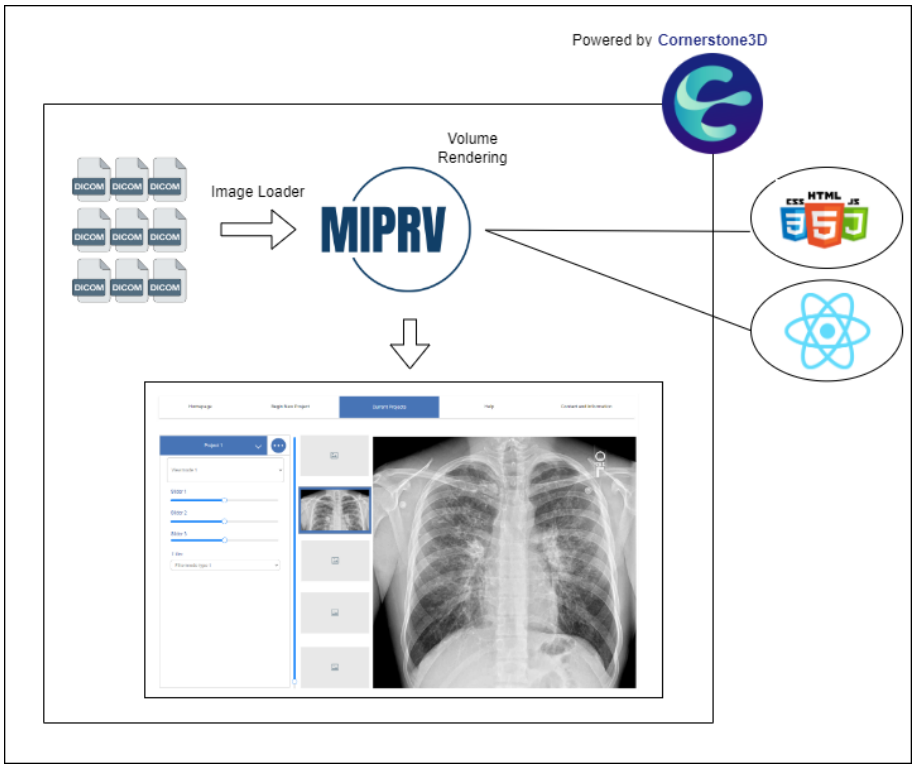


Figure 9: Prototype Website Architecture.

4.3.2 Website

The web platform uses React [20] as the JavaScript framework to build the UI (User interface) and manage UI components of the web platform's Front-End. Using the CornerstoneWadoImageLoader the web platform allows the user to import their own DICOM files directly from the computer and view them in a 2D online format as planned as illustrated in Fig. 10.

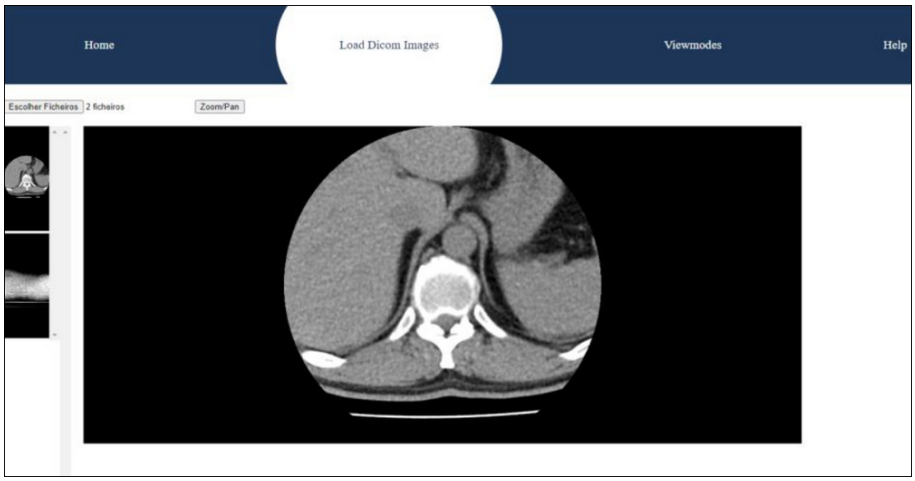


Figure 10: Uploading Page Of The Web Platform.

Once the images have been uploaded, they will be displayed on the center of the screen and can be scrolled through and enlarged individually. Upon completion of the loading process, either by button press or by directly accessing the Viewing tab, the user is redirected to the viewing page. Here, DICOM files are displayed in viewports using Cornerstone3D. The viewing page uses the Cornerstone3D viewports to organize the files, enabling seamless navigation through an array of DICOM images. Users can scroll through the images using the mouse wheel and also zoom in on each individual DICOM image for increased clarity and detail upon double-clicking and using the mouse scroll button. These viewports are positioned in the center of the screen, similar to the DICOM files loaded on the loading page, as depicted in Fig. 11.

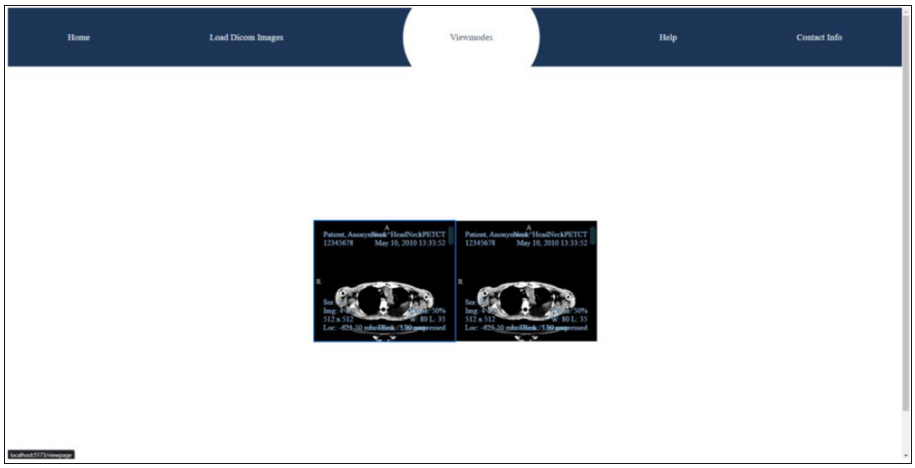


Figure 11: Viewing Page Of The Web Platform.

4.3.3 Issues and solutions

An issue arose where the images sent to the viewports were no longer loading correctly. Attempts were made to resolve this by uploading DICOM files to the web through the Identisoft website. The files were then loaded using the CornerstoneWadoImageLoader and displayed directly in the viewport. However, these attempts proved unsuccessful. The viewport would load the images from the Cornersonte3D database but not files uploaded directly from the machine or other areas of the web. The previously mentioned loader has since been deprecated by the developers, who have moved on to dicom-image-loader. Additionally, the DICOM files provided as examples on the Cornerstone3D website also ceased to function, halting further development of the platform.

Using the OHIF platform, which also uses Cornerstone3D for its rendering also relayed a similar issue, because user-uploaded files are not stored in the browser cache for security reasons reloading the page causes the files to be lost and the renders to not load.

4.4 Conclusion

Despite its infancy, the Cornerstone3D library shows promise in advancing 3D viewing capabilities compared to its previous iteration. It distinguishes itself from standalone viewers by offering a fully online platform, eliminating the need for application downloads and installations, while striving to maintain the performance standards expected of standalone DICOM viewers.

Through the development of the Web platform the performance and loading speeds of the application were fast and moving through the slices of a 3D render was done without reloading. The platform uses the Cornerstone3D framework to process DICOM Files successfully and allows for their viewing in a 2D state, using CornerstoneWadoImageLoader and Cornerstone3D's viewports for stacking these images, the user can scroll through the DICOM files in order.

The platform's development was not completed, not adding the various tools to manipulate and edit the files that were loaded, and not finishing the 3D rendering process.

The platform can benefit from swapping the image loader to the newer option now

used by Cornerstone3D as the CornerstoneWadoImageLoader has since been deprecated, begin user testing and finally provide an online demo freely available. With the issues found and the discovery of other tools already capable of displaying 3D reconstructions of DICOM files, the dissertation moved to compiling several web DICOM viewer technologies into a survey, testing and comparing each viewer's features, gathering the available online demos, creating a performance comparison, and highlighting the strengths and weaknesses of the available, which will be presented in the next chapter.

Chapter 5

DICOM Viewer Analysis

This chapter presents a comprehensive analysis of various DICOM viewers, focusing on their performance, usability, and feature sets. The analysis is structured into two main sections: the first section details the metrics used for the evaluation and comparison of the DICOM viewing software, and the second section surveys existing DICOM viewers, comparing their performance based on these metrics. This chapter aims to provide a detailed understanding of the strengths and weaknesses of different DICOM viewers, guiding users in selecting the most appropriate tool for their needs.

5.1 Metrics

This section goes over the metrics used for the evaluation and comparison of the DICOM viewing software. The choice of metrics for evaluating DICOM viewing software is crucial in assessing the accessibility and functionality of such tools. The acquisition of these metrics involved a thorough review of scientific articles, surveys conducted by other researchers, and extensive online searches [41, 42, 43, 45, 46, 47]. The selection covers a broad range of aspects, ensuring an extensive evaluation. A total of 28 metrics were chosen to provide a comprehensive survey of DICOM viewing software. These metrics are separated into 4 categories, each with multiple sub-categories. The first category, accessibility, covers aspects such as registration, platforms, documentation, contact information, and whether the viewer is free. The second category, interface, assesses the available load options and cloud storage features. The third and fourth categories focus

on 2D and 3D viewing, respectively, with subcategories covering their respective capabilities.

5.1.1 Accessibility

Nine accessibility metrics were set that define how readily available the tool is and if useful information for the software is made available.

M1 – Registration Required

If the software/tool requires the user to go through a registration and login process to begin utilization.

M2 - Free / open-source

If the software is available to be used for free by any user, software with free trials will be considered paid.

M3 – Standalone Version

Some of the surveyed applications contain both a web version and a version that runs standalone on the user's computer.

M4 – Multi-platform

Certain programming languages, like Java, are platform-agnostic. Platform-independent software can run on standalone systems, using Java runtime environments (JRE), or be delivered to client systems via Java Web Start.

M5 – Mobile

If the DICOM viewer offers a suitable graphical user interface (GUI), medical images can be viewed on mobile devices such as smartphones or tablets.

M6 – Documentation

Comprehensive written documentation for the software, such as manuals, is accessible.

M7 – Mail

A mailing list is provided to offer support through email correspondence.

M8 – Forum

A web-based forum is available to provide support.

M9 – Wiki

A user-accessible wiki webpage is provided for reference and assistance.

5.1.2 Interfaces

Four interface metrics were set, used to facilitate communication with other systems required for the loading of DICOM files.

M10 – Load from local files/folders

Allow the loading of content directly from the machine via. DICOM files stored in the computer or of zip files containing them.

M11 – Load from URL

Web protocols such as web access to DICOM objects (WADO) service allow a system to provide access to DICOM objects to other systems directly from the web.

M12 – Parameter Transfer

Parameter calls are employed to directly transfer data, such as settings, upon invoking a software application. For instance, parameter calls can be used to convey contextual information.

M13 - Cloud-Based Storage

The DICOM files/projects are stored in a cloud web server for usage at later dates without the need to have the original DICOM files on the computer.

5.1.3 2D Viewing

Eight 2D viewing metrics were set, they go over the viewing functionality and supported features of the various viewers.

M14 – Scrolling

Mouse scroll allows users to move to the next or previous image, zoom in and out, or swap view mode by simply scrolling with the mouse wheel.

M15 – Metadata

The header viewing functionality involves processing and showing the metadata of DICOM objects.

M16 – Overlay Information

Relevant data should be rendered as an overlay in the display window.

M17 – Windowing

The brightness and contrast of the displayed image are controlled via windowing.

M18 – Pseudo Colors

Pseudo-color look up tables (LUT) convert image gray values to pseudo-colors.

M19 – Measurements

Measurements allow the drawing of content (e.g., lines) as well as the evaluation of geometric figures in the image such as distances or angles.

M20 – Annotations

Image viewing results (e.g., measurements, text annotations) should be saved for subsequent use.

M21 - Histogram

A graphical representation of the distribution of pixel intensity values within the medical image.

5.1.4 3D Viewing

Seven 3D viewing metrics were set, they go over the 3D viewing capabilities of the viewers.

M22 – MPR (Multi-Planar Reconstruction)

Multi-Planar Reconstruction refers to the process of reconstructing two-dimensional images in multiple planes from a three-dimensional dataset.

M23 – Crosshairs

Mouse interaction with the MPR viewports allows for on-click selection of the slice the user wants to see.

M24 – Secondary Reconstruction

Medical volume data is often collected along a single body axis.

M25 – Scroll Through

Moving inside through the 3D render can sometimes provide a better view of a particular part of the file.

M26 – Volume Rendering

Volume rendering is a visualization technique used in medical imaging to create 3D images from a series of 2D medical images.

M27 – Transfer Function

Transfer functionality can be used to map image voxel grey values to tissue opacity values.

M28 – Surface Generation

The creation of three-dimensional (3D) surfaces or models from medical imaging data. Allowing the user to visualize organs, bones, or other structures in a more lifelike and spatially accurate manner.

5.2 DICOM Viewer survey and performance comparison

The survey ascertains whether each DICOM Viewer tool supports or does not support each of the pre-established metrics. A positive sign (+) will denote affirmative support, while a negative sign (-) will indicate a lack of support. To establish the presence or absence of each feature within the applications they were tested, their documentation was meticulously reviewed, and the provided sample projects were examined. Should any of these methods confirm the existence of the feature, it is noted as a positive (+) in the survey. If all three methods fail to substantiate the feature's presence, it is designated negative (-). The survey is displayed in Table 3.

Table 3 indicates that most viewers while relatively feature complete in the 2D viewing area, lack 3D viewing capabilities for DICOM files, with most only supporting MPR with Crosshairs. Slice Drop is an exception, meeting most of the 3D viewing metrics, but lacking most of the 2D ones. OHIF was the highest-scoring viewer in the survey. However, it's

Table 3: Web DICOM Viewer Survey.

		OHIF	MED3WEB	Papaya	DWV	Weasis	IMAIOS	PostDICOM	FViewer	ODV	Sdrop	Oviyam	MedDream
Accessibility	No Registration	+	+	+	+	+	+	-	+	+	+	-	+
	Free / open-source	+	+	+	+	+	+*8	-	+*8	+	+	+	+*8
	Standalone Version	+	+	+	-	+	-	-	-	-	-	-	-
	Multi-Platform	+	+	+	+	+	+	+	+	-	+	+	+
	Mobile	+*1	+*3	-	+	-	+	+	+	-	+	+*1	+
	Documentation	+	+	+	+	+	-	-	-	-	-	-	+
	Mail	+	-	-	+	+	+	+	-	-	-	+	+
	Forums	+	+	+	+	-	-	-	-	+	+	+	-
	Wiki	-	-	-	+	-	-	-	-	-	-	-	-
Interface	Load from File	+	+	+	+	+	+	+	+	+*4	+	+	+
	Load from URL	+	+	+	-	+	+	+	+	+*4	-	+	-
	Cloud-Based Storage	+*2	-	-	-	-	-	+	-	-	-	-	+
2D Viewing	Scrolling	+	+	+	-	+	+	+	-	+	+	+	+
	Metadata	+	+	-	-	+	-	+	+	-	-	+	+
	Overlay	+	+	+	+	+	+	+	+	-	-	+	+
	Windowing	+	+	+*5	+	+	+	+	+	+	+	+	+
	Pseudo Colors	-	-	+	-	+	-	+	+	-	+	-	+
	Measurements	+	+	+	+	+	+	+	+	-	-	+	+
	Annotations	+	+	+	+	+	-	+	+	-	-	+	+
	Histogram	+	+	-	-	-	-	-	+	-	-	-	+
3D Viewing	MPR	+	+	+	-	+	+*6	+	-	-	+	-	+
	Crosshairs	+	+	+	-	+	-	+	-	-	-	-	+
	Secondary Reconstruction	+ *7	-	-	-	-	-	+	-	-	-	-	-
	Scroll Through	-	+	-	-	-	-	-	-	-	+	-	-
	Volume Rendering	+ *7	+	-	-	-	-	+	-	-	+	-	+
	Transfer Function	+ *7	+	-	-	-	-	+	-	-	+	-	+
	Surface Generation	+ *7	+	-	-	-	-	-	-	-	+	-	+

*1 Works but it's not officially supported, it has layout issues such as image display boxes being too small or too large;

*2 Has cloud saving but requires OHIF approval for the study to be stored;

*3 Mobile has limited functionality having lost some of the features;

*4 No longer works on modern machines

*5 The Sample Application comes only with 4 filters having no brightness adjustment on the 2D images;

*6 The mode is available but failed to load local DICOM files when testing

*7 Not available for locally loaded DICOM files only for studies in the cloud

*8 Free but not open-source.

important to note that its 3D viewing features are only accessible with example cases uploaded to the cloud by the developers, not locally loaded DICOM files. Taking the previous into account, OHIF met 24 out of the 27 defined metrics, MED3WEB met 22, MedDream met 21, and PostDICOM met 18, failing most of the accessibility metrics. Following these, Papaya and Weasis each meet 17 metrics, followed by SliceDrop with 14. Oviyam, DWV, and Fviewer each meet 13, and finally, IMAIOS and ODV meet 12 and 7 metrics respectively. If 3D viewing is not included, the top performers are OHIF and MedDream, meeting 16 out of 18 metrics, followed by Med3Web with 15, and Weasis with 13.

5.2.1 Measuring Method

The forthcoming performance comparison uses a free and public dataset featuring 517 DICOM images, accessible on the website Medimodel [49], showcasing a human skull afflicted with class 3 malocclusion. This dataset can be seen in Fig 12 which depicts a 3D visualization of the DICOM files in the dataset.

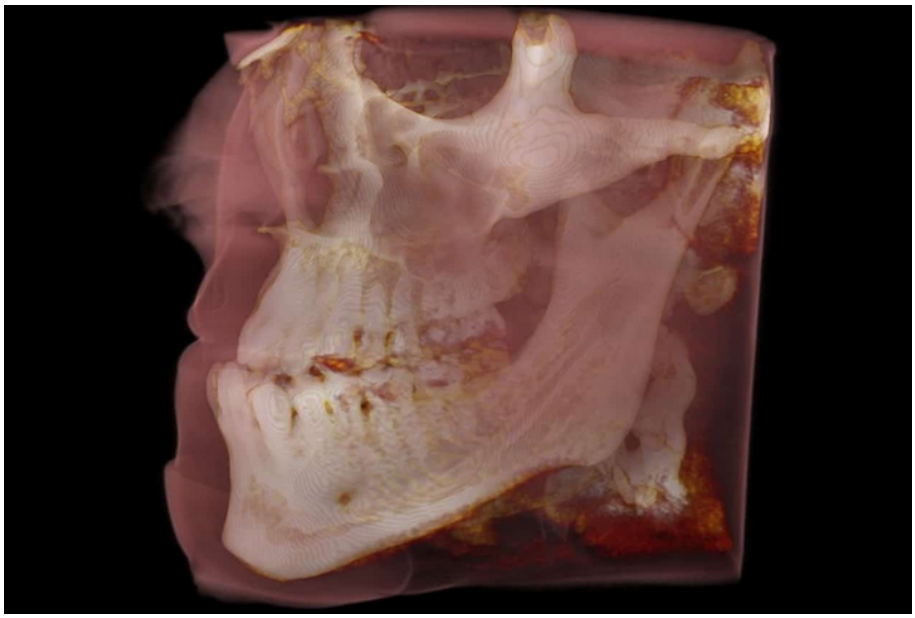


Figure 12: Visualization of the Class 3 malocclusion DICOM dataset.

The assessments will commence by loading the images into the designated software, followed by the initiation of the 3D rendering process. These tests will be performed on five distinct computers, which will be named, computers A, B, C, D, and E. Their

specifications can be found in the appendix file C, with computers A and B being modern machines with Windows 11, C an older computer with Windows 10, and D, and E being Mac computers.

The tests were conducted on 4 different browsers including Google Chrome, Microsoft's Chromium-based browser known as 'Edge,' Firefox, and Safari. All tests were run on the computers with any other nonessential applications closed to optimize and streamline performance, allowing for reliable comparisons. Tests were only conducted on viewers with an online example ready for usage; viewers with only locally-run servers available were not considered for the performance comparison. The viewers OHIF, Med3EWeb, DWV, IMAIOS, PostDICOM, Fviewer, Slice Drop, and the DICOMLibrary+MedDream combo qualified for the performance comparison.

To gauge the duration of the software's task completion, the computer screen was recorded using OBS (Open Broadcaster Software). The time intervals between the initiation of the loading process (once the files are selected, and the uploading process begins), and the display of the 2D image on the screen was measured. Additionally, the time intervals between pressing the 3D rendering button and the display of the render on the screen were recorded. Each table in this section features browsers in the first column, followed by a column for each computer. If a test is not conducted on a specific browser and computer, it is marked with (-); if a test fails to load the images, it is marked as (Fail), otherwise, the time it took to finish is displayed in seconds.

At least two tests were conducted per viewer, for both 2D and 3D (when available). The first test was conducted on a computer with its cache cleared or with the browser in private mode. The second test involves accessing the website for at least the second time, allowing the usage of the browser's cache.

The tests should consider an error margin of around 300 milliseconds for the human reaction time [50] as the recordings still need to be paused for the time to be taken. The fastest 2D and 3D (when available) test times from tests 1 and 2 are underlined on each table, with a margin of error of 300ms from the quickest result.

5.2.2 PostDICOM Testing

PostDICOM requires the user to first upload their DICOM files to their cloud, which, depending on the user's upload speed, can take hours to upload larger datasets. The time taken to upload the files was not counted towards loading times or display times but was taken into account when the comparison with other viewers was made.

Following the successful loading of the 2D render, the time between the click of the 3D render button and the display of the 3D render was measured using the assistance of recording software. The time intervals can be seen in Table 4 in the 3D Rendering tests.

For the 2D Rendering times, the duration between the click of the upload file button and the conclusion of the loading process was taken using the assistance of recording software. The time for each test taken can be seen in Table 4 in the 2D rendering tests.

An extra test was done to confirm the abnormal 2D rendering results in Test 1 on (Firefox - C) and (Edge - A). Using the browsers in private mode on the C computer and using browser cache for computer A. The results were 04.53s C on Firefox and 04.08s A on Edge.

Table 4: PostDicom Test Results.

	Computer				
	A	**B**	**C**	**D**	**E**
3D Rendering Test 1					
Firefox	05.71s	06.54s	07.05s	-	-
Edge	06.82s	07.32s	09.03s	-	-
Chrome	05.89s	05.90s	07.34s	05.61s	06.60s
Safari	-	-	-	Fail	08.88s
3D Rendering Test 2					
Firefox	05.51s	06.31s	07.49s	-	-
Edge	06.86s	05.56s	06.12s	-	-
Chrome	05.70s	06.40s	04.23s	05.52s	06.16s
Safari	-	-	-	Fail	06.61s
2D Rendering Test 1					
Firefox	06.28s	06.52s	01.99s	-	-
Edge	04.15s	08.12s	02.61s	-	-
Chrome	03.05s	03.08s	15.87s	02.86s	04.21s
Safari	-	-	-	Fail	05.10s
2D Rendering Test 2					
Firefox	01.04s	03.27s	00.98s	-	-
Edge	04.11s	07.33s	02.15s	-	-
Chrome	00.87s	01.86s	15.96s	01.07s	01.66s
Safari	-	-	-	Fail	01.99s

The extra test confirmed that although the faster loading experienced on computer C with Firefox was not as quick as initially observed, it was still faster than the loading speeds on other computers. Additionally, despite having better hardware, computer A performed worse than computer C on Edge and Firefox in the 2D tests.

Overall, based on the findings in Table 4, it can be concluded that Chrome is the best-performing browser for the stronger computers on postDICOM, exhibiting similar results to Firefox on 3D viewing and the best results in 2D viewing. For less powerful computers, Edge and Firefox emerge as better alternatives. Mac systems should also consider using Chrome when using postDICOM.

5.2.3 OHIF Testing

OHIF does not currently allow for the 3D rendering of files locally. The page needs to be refreshed to show the 3D rendering, causing the web app to lose access to data. It's not possible to download their available studies stored on the cloud and load them on other DICOM viewers. As a solution, to still allow comparison with the other viewers, a study that uses 642 instances designated "CT CHEST WITH IV COUNT" was picked as a replacement. This study will be loaded and rendered in 2D and, afterward, the hanging protocol to show the 3D Render (&hangingprotocolId=mprAnd3DVolumeViewport) will be added. The page will be reloaded, and the time interval will be measured between the reload and the finish of the 3D render. It is important to note that OHIF performed the 2D and 3D rendering processes simultaneously as there is no way to separate the two currently.

Since OHIF loads the files, and then, gives the option to start the viewer, the 2D rendering tests were separated into two times, the time to load the files into the software, and the time to display the files on the screen, where [Result = (Loading time + Display time)]. The loading time was recorded between the click of the upload files button and the start of the viewer, while the display time was recorded between the click of the view button and the display of the 2D render. The time intervals for the first test are indicated in the appendix in D.2 displayed in Table 11.

The time acquisition process can be seen in Figure 13 where the loading time is highlighted by the red arrows and the display time is highlighted by the blue arrows.

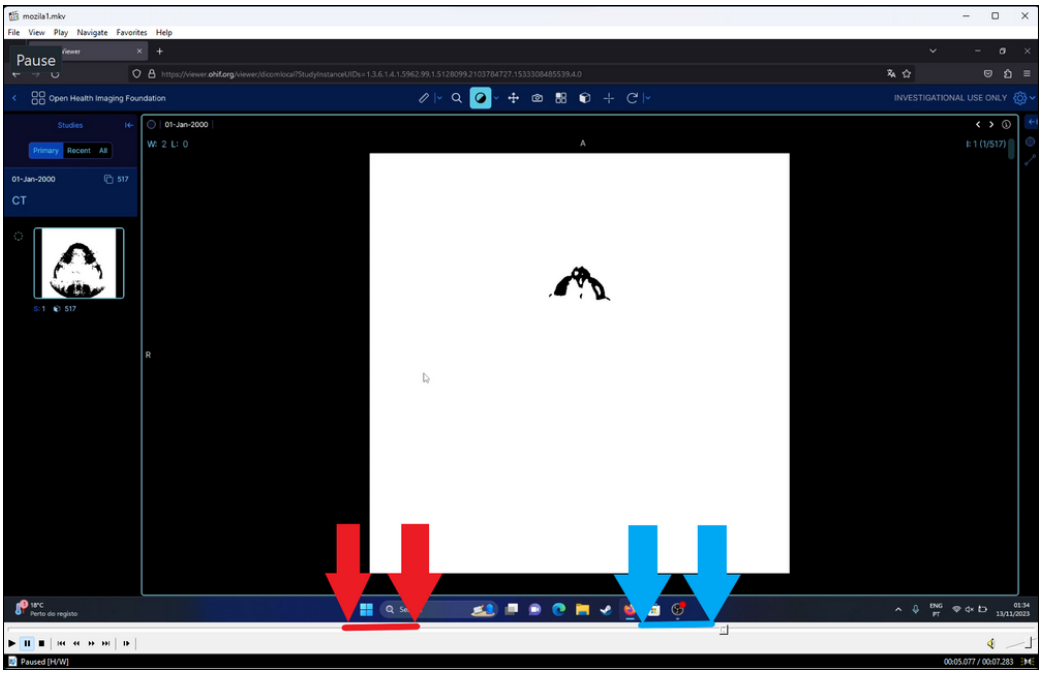


Figure 13: OHIF testing and measuring on the Firefox browser.

The time intervals for the second test are in the appendix on D.2 displayed in Table 12. The full table with all the results can be seen in Table 5.

Table 5: OHIF Rendering Test Results.

	Computer				
	A	**B**	**C**	**D**	**E**
3D Rendering Test 1					
Firefox	02.02s	02.97s	07.05s	-	-
Edge	02.52s	05.04s	09.03s	-	-
Chrome	01.11s	01.51s	03.70s	02.70s	07.34s
Safari	-	-	-	Fail	Fail
3D Rendering Test 2					
Firefox	01.88s	02.34s	07.49s	-	-
Edge	03.07s	03.57s	06.12s	-	-
Chrome	00.93s	00.97s	03.16s	02.69s	04.23s
Safari	-	-	-	Fail	Fail
2D Rendering Test 1					
Firefox	01.18s	00.93s	01.99s	-	-
Edge	01.43s	01.47s	02.61s	-	-
Chrome	00.90s	01.24s	02.56s	02.27s	02.61s
Safari	-	-	-	Fail	Fail
2D Rendering Test 2					
Firefox	00.64s	00.92s	00.98s	-	-
Edge	01.47s	01.17s	02.15s	-	-
Chrome	00.72s	00.64s	01.29s	01.31s	01.62s
Safari	-	-	-	Fail	Fail

Overall, based on the findings in Table 5, Chrome and Firefox perform similarly in the 2D viewing area, with Edge lagging behind the alternatives and Safari not being supported by the viewer. When looking at 3D viewing Chrome performed better in all parameters regardless of machine making it the go-to browser for this platform. Based on Table 11 and 12 in annex D.1, it can be concluded that both the loading time and the display time benefit from the browser cache.

5.2.4 IMAIOS Testing

IMAIOS tests begin with clicking the upload file button and end with the completion of the loading of the last file. The tests can be seen in Table 6.

Table 6: IMAIOS Rendering Test Results.

	Computer				
	A	**B**	**C**	**D**	**E**
2D Rendering Test 1					
Firefox	03.23s	13.59s	20.09s	-	-
Edge	03.54s	246.84s	276.01s	-	-
Chrome	01.82s	12.83s	29.69s	22.37s	28.99s
Safari	-	-	-	46.42s	38.58s
2D Rendering Test 2					
Firefox	03.44s	13.61s	22.04s	-	-
Edge	07.79s	241.89s	437.11s	-	-
Chrome	01.79s	16.06s	29.88s	21.31s	28.96s
Safari	-	-	-	37.50s	36.74s

Based on the data in Table 6, Chrome emerges as the better performer overall, except for Computer C, where Firefox was the faster loader. Chrome should be considered for stronger computers and Firefox for older, or weaker, computers. Additionally, Edge's results were exceedingly slow for any computer other than A.

5.2.5 DWV Testing

The DWV tests begin with clicking the upload file button and end when all the files have finished loading (indicated by the completion of the loading bar).

Table 7: DWV Rendering Test Results.

	Computer				
	A	**B**	**C**	**D**	**E**
2D Rendering Test 1					
Firefox	04.46s	01.30s	12.14s	-	-
Edge	06.57s	08.37s	21.89s	-	-
Chrome	03.67s	01.02s	12.59s	16.02s	15.86s
Safari	-	-	-	Fail	15.47s
2D Rendering Test 2					
Firefox	03.97s	01.34s	13.03s	-	-
Edge	06.68s	08.40s	22.16s	-	-
Chrome	01.03s	01.18s	12.07s	15.43s	15.74s
Safari	-	-	-	Fail	15.04s

In Table 7, Chrome emerges as the faster option overall, but it lags slightly behind Firefox on computer C. Additionally, Edge's performance was slower compared to the alternatives.

5.2.6 FViewer Testing

The FViewer tests begin with clicking the upload file button and end when all the files have finished loading (indicated by the processing wheel stopping).

Additional tests were conducted on computer A on Edge to confirm faster load times compared to the other computers, and on computer C, also on Edge, to confirm slower loading times experienced on this browser compared to the others. The caches were not cleared, and the results were as follows: 27.65s for computer A and 53.41s for computer C.

Table 8: FViewer Rendering Test Results.

	Computer				
	A	**B**	**C**	**D**	**E**
2D Rendering Test 1					
Firefox	Fail	Fail	Fail	-	-
Edge	12.13s	32.09s	26.77s	-	-
Chrome	04.63s	06.26s	08.83s	09.13s	12.91s
Safari	-	-	-	10.08s	13.90s
2D Rendering Test 2					
Firefox	Fail	Fail	Fail	-	-
Edge	31.37s	31.06s	54.61s	-	-
Chrome	05.22s	06.47s	10.63s	08.75s	13.61s
Safari	-	-	-	08.46s	12.05s

Based on the data in Table 8, it is evident that FViewer does not benefit from browser cache usage, as the second results were consistently slower, or, similar across all browsers except Safari. Notably, Safari performed slightly better than Chrome on MAC computers. Firefox failed to run with FViewer on all tested computers. Additionally, Edge's performance was notably slower compared to the alternatives.

5.2.7 Med3Web Testing

Because MED3WEB loads the files before giving the option to choose between 8-bit and 16-bit rendering, the 2D rendering tests were divided into two time intervals: the time to finish file processing (Loading time), and the time from the selection of the processing method to the display of the rendering (Display time). Similar to the procedure used with OHIF, the overall result was calculated as [Result = (Loading time + Display time)]. For consistency, the 16-bit rendering method was selected across all devices and browsers. The time intervals for the first test are provided in the appendix on D.2, displayed in Table 13. The time intervals for the second test are in the appendix on D.2, displayed in Table 14.

After the successful loading of the 2D render, the time between clicking the 3D render button and the display of the 3D render was measured using recording software. The results can be seen in Table 9 on 3D rendering Test 1 and 2.

Accidental inputs were made during the testing of the 3D rendering on device A, specifically while using Edge by clicking the 3D rendering UI. To ensure a fair comparison with other devices, an additional test was conducted on Edge without clearing the browser cache. The result was as follows: 34.21s A Edge.

The full table with all Med3Web's results can be seen in Table 9.

Table 9: Med3Web Test Results.

	Computer				
	A	**B**	**C**	**D**	**E**
3D Rendering Test 1					
Firefox	04.96s	05.56s	17.08s	-	-
Edge	49.21s	36.93s	98.49s	-	-
Chrome	02.73s	03.63s	18.81s	04.58s	05.12s
Safari	-	-	-	Fail	05.52s
3D Rendering Test 2					
Firefox	03.13s	05.29s	19.48s	-	-
Edge	55.78s	36.76s	96.06s	-	-
Chrome	02.54s	03.13s	20.19s	04.25s	04.84s
Safari	-	-	-	Fail	04.51s
2D Rendering Test 1					
Firefox	09.23s	09.89s	28.27s	-	-
Edge	152.19s	155.19s	309.14s	-	-
Chrome	07.42s	08.22s	21.40s	15.56s	15.87s
Safari	-	-	-	Fail	14.04s
2D Rendering Test 2					
Firefox	10.53s	07.42s	23.82s	-	-
Edge	151.97s	156.02s	302.62s	-	-
Chrome	07.40s	14.99s	21.20s	15.29s	15.96s
Safari	-	-	-	Fail	12.85s

Table 9 indicates that Chrome performed the best in both 2D and 3D rendering on Windows machines, although it lagged behind the Safari browser on computer E. Additionally, Edge's rendering times were significantly longer on all three Windows computers, making it nearly unusable.

Based on Tables 13 and 14 in Appendix D.2, it can be concluded that the time needed to load the images is consistently longer than the time to display the images on the screen and that no improvements are found with browser cache.

5.2.8 MedDream Testing

If the user intends to use MedDream with their own study, the viewer requires users first to upload their DICOM files to the DICOMLibrary cloud. Depending on the user's upload speed, the uploading process can take multiple hours for large datasets. The time taken to upload the files was not counted towards loading or display times but was considered during comparisons with other viewers.

After successfully uploading the DICOM dataset to the web, DICOMLibrary generates

links to access and view the files on MedDream. The time interval between opening the link and displaying the files in 2D was measured and is displayed in Table 10 in the 2D tests.

Once the viewer finishes the loading process, the time between clicking the 'MPR - Mist Oblique' button and displaying the 3D render of the dataset was measured and is displayed in Table 10 in the 3D tests."

A third test was conducted to confirm the improved loading times during the second test on the browsers on Computer E. The results showed 10.87 seconds on Chrome and 12.81 seconds on Safari, confirming the improved results with cached data.

Table 10: MedDream Test Results.

	Computer				
	A	**B**	**C**	**D**	**E**
3D Rendering Test 1					
Firefox	14.35s	16.09s	46.10s	-	-
Edge	06.35s	34.16s	Fail	-	-
Chrome	06.34s	07.12s	Fail	-	Fail
Safari	-	-	-	-	Fail
3D Rendering Test 2					
Firefox	14.43s	16.33s	44.32s	-	-
Edge	06.59s	29.89s	Fail	-	-
Chrome	06.01s	07.17s	Fail	-	Fail
Safari	-	-	-	-	Fail
2D Rendering Test 1					
Firefox	02.16s	02.28s	03.18	-	-
Edge	01.58s	02.70s	03.35s	-	-
Chrome	02.37s	02.69s	03.20s	-	36.60s
Safari	-	-	-	-	51.50s
2D Rendering Test 2					
Firefox	01.71s	01.37s	02.67s	-	-
Edge	01.19s	01.76s	02.59s	-	-
Chrome	01.03s	01.62s	02.22s	-	08.64s
Safari	-	-	-	-	15.01s

Table 10 indicates that Chrome performed the best in both 2D and 3D rendering tests. However, it failed to load the 3D render of the file on computer C. Only the Firefox browser was able to obtain a 3D render on computer C. The 3D rendering tests ran for fifteen minutes on computer C before being marked as failed. Notably, the loading bar reached halfway around the one-minute mark and did not progress further on both browsers.

5.2.9 Drop Slice Testing

Slice Drop failed to properly display the dataset on all computers and browsers in both 2D and 3D, instead displaying the result depicted in Figure 14. Although a smaller test using a twenty-file dataset led to the display working correctly, the discrepancy in size was too large, resulting in the testing being marked as unsuccessful.

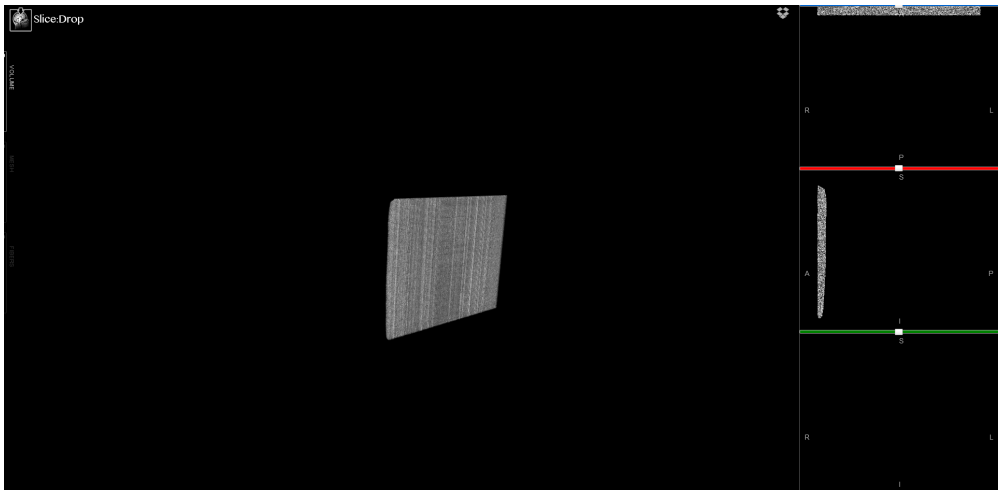


Figure 14: 3D Visualization of the Class3Malocclusion Dataset on Slice Drop.

5.3 Results and Discussion

The performance tests on various DICOM viewers yielded insightful results across different rendering scenarios, browsers, and operating systems. PostDICOM saw the best results across all viewers for local 3D rendering on computer C proving an excellent choice for weaker computers. On this viewer, Firefox and Chrome exhibited similar performance, while Edge lagged slightly behind. In the area of 2D rendering in PostDICOM, a substantial improvement was observed during the second rendering, particularly noteworthy on Firefox for Computer C. Interestingly, Chrome outperformed Firefox on more powerful machines (A and B) but struggled on the older machine (C) in the 2D rendering tests. Additionally, newer versions of Safari are required to use this browser on PostDICOM specifically versions above 15.1.

Regarding the OHIF viewer, the 3D rendering interface, although not integrated into the user interface, can be accessed by adding a ”hanging protocol” to the project link. However, issues surfaced when using locally loading DICOM files, disrupting rendering

upon the reloading of the page. Developers have communicated future implementation, underlining that "An interface for 3D rendering" is planned for the future. The 3D rendering tests were conducted using a different study with 642 instances instead. The results showed that, even with a larger study, OHIF remains the fastest available web browser for 3D rendering. While no significant differences were noted between Firefox and Chrome, Edge consistently lagged slightly behind on all computers. Additionally, Safari for iOS systems does not work with the current version of OHIF, regardless of the Safari version. Display times for OHIF proved consistently superior to all other viewers, especially considering that files do not need to be uploaded to the OHIF cloud before usage.

In the case of the DWV Viewer compatibility issues arose, with DWV failing on Safari 13.1.2. Additionally, excluding the PNG file present in the dataset was necessary for proper folder loading on Devices A and B. This issue did not occur on Computer C, suggesting a connection to Windows 11. The viewer lags behind and offers less functionality than more modern counterparts such as OHIF.

IMAIOS Viewer showed large discrepancies during the loading of DICOM files. Notably, Edge exhibited better performance on computer A when compared with all other machines which had very long load times. However, Edge still fell behind Chrome and Firefox. A combination of a desktop computer with Windows 11 and very good hardware proves essential for getting good results from this viewer. It's important to note that even the best results in IMAIOS lag behind OHIF.

FViewer performed worse than IMAIOS and DWV on Windows but better than these viewers on iOS. However, it still performed worse than OHIF and PostDICOM across all parameters. Additionally, this viewer separates all images into individual tabs, making it less convenient for users to scroll through.

Med3Web's 3D rendering on computer A initially displayed slower results due to an accidental input. However, without this input on test 3, the results mirrored those on computer B. Older versions of Safari (13.1.2 or below) do not work with the software, and Edge had poor results. Chrome and Firefox, on the other hand, performed similarly with the content loading reasonably fast, however, even on stronger machines, a one to two-second delay was observed when altering or moving the render on the 3D side of the

app. The 3D rendering on Med3Web proved slower than that provided by PostDICOM and OHIF. However, it's important to note that Med3Web is free and already offers the interface for volume rendering, unlike OHIF. In contrast to alternative viewers, Med3Web allows users to selectively determine the quality of the rendering process and opt for accelerated 3D rendering, having more customization options.

The DICOMLibrary and MEDDream combo proves to be a good free competitor to PostDICOM, providing most of the same features, including cloud storage, with better overall performance in 2D and slightly worse in 3D. However, on the Windows 10 computer, the viewer failed to load the 3D render of the dataset on Edge and Chrome. The loading bar would finish, but the display would not render. The test ran for over 30 minutes until it was closed. This viewer could prove a valuable alternative for users seeking better performance than Med3Web (although the loading speed for 3D will be slightly higher, the render will not be prone to freezing when altered), and are willing to overlook the time required for file uploads.

Notably, the evaluation revealed that MEDDream, PostDICOM, and OHIF demonstrated consistent improvements within the second 2D test, leveraging the advantages of browser caching. These viewers showcased better time intervals on the second 2D rendering test on most computers and browsers, enhancing efficiency over multiple sessions. On another note, the 3D rendering tests did not show a consistent improvement in time on the second test.

Hardware-wise, computer A, the computer with the best hardware, outperformed others in all browsers and viewers, with a single exception on PostDICOM, where computer C, arguably the weakest, had the best result. The improvement in performance from computer A to the others varied greatly, ranging from less than a second to minutes, depending on the viewer. The viewers most impacted by weaker hardware were IMAIOS and Med3Web. Conversely, the viewers less impacted by hardware were OHIF and PostDICOM.

As Chrome was the only browser that was run across all computers and did not fully fail any single viewer, it can be used for an overall comparison of all the viewers. This comparison can be seen in Fig. 15 for 2D rendering and Fig. 16 for 3D rendering.

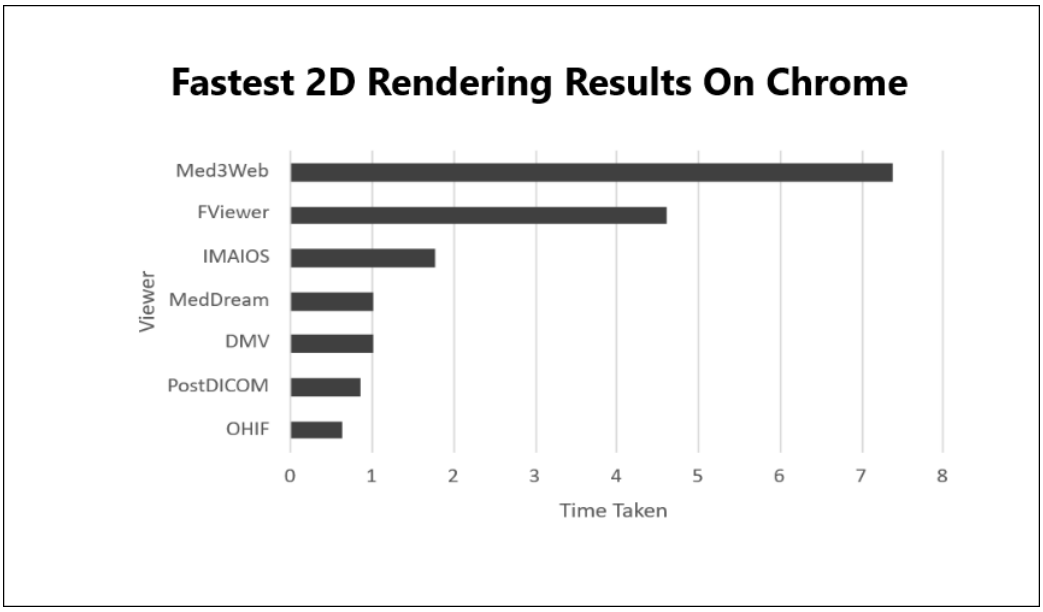


Figure 15: Chrome 2D rendering comparison chart.

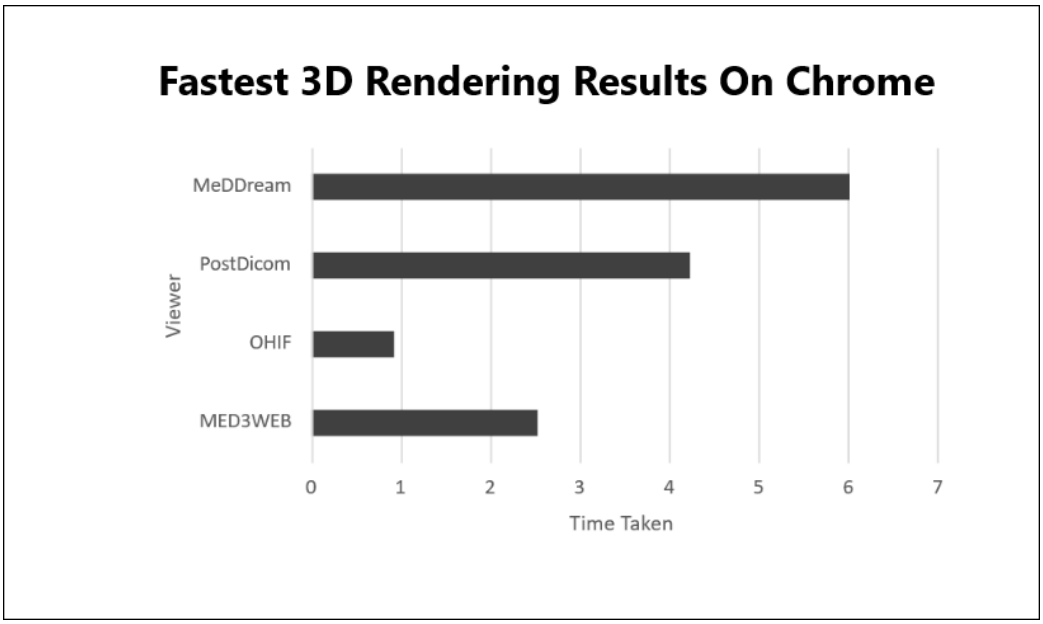


Figure 16: Chrome 3D rendering comparison chart.

From Fig. 15 and Fig. 16 it can be concluded that OHIF is the winner in terms of fast performance, however, it's important to keep in mind the viewer's setbacks in terms of 3D viewing, which are not present in the other two.

5.4 Conclusion

The comprehensive evaluation of various DICOM viewers has provided valuable insights into their performance and features. Among the viewers tested, OHIF and PostDICOM emerged as the fastest among those with web apps available, with MEDDream lagging slightly behind in the 3D viewing area but still showcasing efficient 2D load times. OHIF, despite having its 3D rendering interface in development, demonstrated superior efficiency in the features it has implemented. Med3Web exhibited slower performance compared to OHIF on both 3D and 2D rendering, and compared to MEDDream and PostDICOM, it had slower 2D rendering speeds and lag when making alterations to the 3D render. However, faster rendering methods can be selected to increase speed and responsiveness. Additionally, unlike PostDICOM and MEDDream, it doesn't require users to upload their files to the cloud before usage.

PostDICOM, on the other hand, comes with numerous features readily available and stands out as a versatile option with the second-best performance overall, and the best performance on weaker computers. However, it comes with a notable caveat – it is a paid service. MEDDream serves as a viable alternative that also offers a competitive number of features, including cloud storage. The inclusion of cloud access further extends their functionality but requires file uploads, which will be time-consuming, and depending on file size, and network upload speed, could add hours to the time needed for users to finally view the files. IMAIOS, DWV, and FViewer, on the other hand, demonstrated slower performance compared to OHIF, MEdDream, and PostDICOM. As they only offer 2D features, OHIF is the better option, as the other viewers need to improve in terms of speed and responsiveness to remain competitive.

Browser choice significantly influences the display speed, with Chrome and Firefox being the faster options. Furthermore, Windows machines generally outperformed their Mac counterparts. However, Safari performed similarly to Chrome when supported by the viewer. This insight is crucial for users seeking to optimize their viewing experience based on browser preferences.

In summary, the choice of a DICOM viewer depends on specific user requirements, preferences, and considerations. OHIF, with its speed, presents a compelling option for

users prioritizing efficiency and performance. PostDICOM and MEDDream offer cloud access (via DICOMLibrary for MEDDream's case) and are feature-complete viewers, making them the best choice for projects that require multiple sessions, provided the time needed for uploading is not a concern. For 3D rendering, Med3Web may be suitable for users willing to tolerate occasional stutters when interacting with their render, as a trade-off for avoiding long uploads to a cloud. When focusing on 2D viewing and fast results, OHIF emerges as the better option, offering features comparable to Weasis and PostDICOM, along with the fastest rendering available online for free.

The next chapter will present the conclusion derived from all the work accomplished in this project.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The field of medical imaging has witnessed remarkable advancements in recent years, driven by the integration of digital technologies and the development of sophisticated DICOM viewers. In this dissertation, a solution using the innovative library Cornerstone3D was created, and a comprehensive analysis of various DICOM viewers was conducted, evaluating their performance and features to provide valuable insights for medical professionals, researchers, and developers.

Cornerstone3D, along with the technology to convert DICOM files into viewable content, was thoroughly explored during the early development stages of the web application. While the software is still in its infancy and undergoing significant changes as it evolves, the 2D viewing area has proven to be fast, stable, and well-documented. This aspect is particularly beneficial for developers who wish to experiment with 3D rendering but require a solid foundation in 2D visualization.

From the first set of objectives set for the development of the web platform, the first three objectives set for the development of the web platform were fully met as the web app is capable of loading and viewing DICOM files directly from the machine in real-time as they load, the following objectives, however, were only partially met as while they work when ran separately (Using the images and tutorials provided by Cornerstone3D), once integrated with the application the files are not loaded. The app was not finished; thus, the volume rendering, image fusion, and crosshairs were not integrated, and user

testing was not done. Despite the challenges associated with developing a robust 3D rendering interface, the progress made thus far is promising, and with continuous updates on the Cornerstone3D framework, developing the web app's 3D Visualization areas is still possible.

Furthermore, this dissertation conducted a comprehensive analysis of various DICOM viewers, focusing on their performance and features. Through rigorous testing and evaluation, valuable insights into the strengths and limitations of each viewer were obtained.

Significant variations in the performance of DICOM viewers across different browsers, operating systems, and hardware configurations were observed. OHIF and PostDICOM emerged as the fastest viewers in terms of rendering speed, with MEDDream lagging slightly behind in 3D viewing but showcasing efficient 2D load times.

The feature availability of the various viewers was tested and detailed with PostDICOM and MEDDream standing out as feature-complete viewers, offering a wide range of functionalities including cloud storage, and advanced visualization tools. However, they come with the caveat of requiring file uploads, which can be time-consuming.

Considering this, the second set of objectives for the DICOM viewer comparison was fully met.

In conclusion, the choice of a DICOM viewer depends on specific user requirements, preferences, and considerations. As mentioned in Chapter 5.4 OHIF emerges as the option for users prioritizing efficiency and performance, while MEDDream and PostDICOM offer viable alternatives with competitive features and cloud access. Continued innovation and collaboration in this field will further advance the capabilities of DICOM viewers and contribute to improved healthcare outcomes for patients worldwide.

This dissertation contributes to the existing body of knowledge by providing a comprehensive analysis of DICOM viewers and offering actionable insights for medical professionals, researchers, and developers.

6.2 Future work

Future work in the web app area could focus on refining the 3D rendering capabilities of the application, addressing any existing limitations or performance bottlenecks, and beginning user testing. Collaborative initiatives with clinicians and researchers would also be invaluable for validating the utility of the 3D rendering features in real-world healthcare settings.

Furthermore, as new tech enters the field, in the future, performance comparisons should try to reach similar results to OHIF but without the caveat of lacking the option to visualize a 3D render of files loaded locally. A viewer that allows for uploading files to the cloud, but without it being a requirement, would be useful for keeping important work saved but also allow for quick viewing of less important files.

References

[1] Manuel Lederman. "The early history of radiotherapy: 1895–1939". In: *International Journal of Radiation Oncology* Biology* Physics* 7.5 (1981), pp. 639–648.

[2] National Electrical Manufacturers Association et al. *Digital Imaging and Communications*. The Association, 1985.

[3] Charles E Kahn et al. "DICOM and radiology: past, present, and future". In: *Journal of the American College of Radiology* 4.9 (2007), pp. 652–657.

[4] PostDICOM B.V. PACS integrated online DICOM Blog - How DICOM Viewers Facilitate Patient-Centric Care in the Digital Age. Last accessed on May 22 2023. URL: https://www.postdicom.com/en/blog/how-dicom-viewers-facilitate-patient-centric-care.

[5] Erik Ziegler et al. *Open Health Imaging Foundation Viewer: An Extensible Open-Source Framework for Building Web-Based Imaging Applications to Support Cancer Research*. Version 1.0.0. DOI: 10.1200/CCI.19.00131. URL: https://github.com/OHIF/Viewers.

[6] David E Avison et al. "Action research". In: *Communications of the ACM* 42.1 (1999), pp. 94–97.

[7] John Venable. "The role of theory and theorising in design science research". In: *Proceedings of the 1st International Conference on Design Science in Information Systems and Technology (DESRIST 2006)*. Citeseer. 2006, pp. 1–18.

[8] Vijay K Vaishnavi. *Design science research methods and patterns: innovating information and communication technology*. Auerbach Publications, 2007.

[9] Peter Mildenberger, Marco Eichelberg, and Eric Martin. “Introduction to the DICOM standard”. In: *European radiology* 12 (2002), pp. 920–927.

[10] Bahar Mansoori, Karen K Erhard, and Jeffrey L Sunshine. “Picture Archiving and Communication System (PACS) implementation, integration & benefits in an integrated health system”. In: *Academic radiology* 19.2 (2012), pp. 229–235.

[11] David Flanagan. *JavaScript: o guia definitivo*. Bookman Editora, 2012.

[12] Gavin Bierman, Martín Abadi, and Mads Torgersen. “Understanding typescript”. In: *European Conference on Object-Oriented Programming*. Springer. 2014, pp. 257–281.

[13] Cornerstone Team. *CornerstoneJs, 2022 [Official Website] documentation*. 2022. URL: `https://www.cornerstonejs.org/docs/getting-started/overview` (visited on 05/06/2024).

[14] hongliang666/med3web. Med3Web Viewer. Last accessed on November 12 2024. URL: `https://github.com/hongliang666/med3web`.

[15] rii-mango/Papaya. Papaya Viewer Last accessed on September 22 2024. URL: `https://github.com/rii-mango/Papaya/wiki/Configuration`.

[16] ivmartel/dwv. DicomWebViewer Last accessed on November 13 2024. URL: `https://github.com/ivmartel/dwv`.

[17] Nicolas Roduit. nroduit/Weasis. Weasis Viewer Last accessed on September 20 2023. URL: `https://github.com/nroduit/Weasis`.

[18] IMAIOS Team. IMAIOS DICOM VIEWER Last accessed on November 12 2023. URL: `https://www.imaios.com/en/imaios-dicom-viewer`.

[19] PostDICOM B.V. PACS integrated online DICOM viewer. Last accessed on September 18 2023. URL: `https://www.postdicom.com/`.

[20] Crypto V. VOXELX — THE ONLINE MEDICAL DICOM-PLATFORM April 22 2018. URL: `https://medium.com/@cryptoverygood/voxelx-the-online-medical-dicom-platform-57c4d8f805ba`.

[21] Fviewer Online Cloud Viewer. Last accessed on November 11 2023. URL: `https://www.fviewer.com/`.

[22] 2023 Awsultan Jotacamara. Open Dicom Viewer. Last accessed on November 2. URL: https://sourceforge.net/projects/opendicomviewer.

[23] 2024 Osimis Official Website. Last accessed on March 15. URL: https://www.osimis.io/.

[24] 2024 Rowen James. Dicombinator. Last accessed on March 15. URL: https://github.com/jamesrowen/dicombinator.

[25] 2024 Posthuma Jorrit. Hida Dicom Viewer. Last accessed on March 28. URL: https://github.com/JorritPosthuma/Hida.

[26] 2024 SliceDrop Interactive Viewer For Medical Imaging Data. Last accessed on March 15. URL: https://github.com/slicedrop/slicedrop.github.com.

[27] 2024 Oviyam Official Website. Last accessed on March 15. URL: http://oviyam.raster.in/oviyam2.htm.

[28] 2024 DicomLibrary Official Website. Last accessed on March 15. URL: https://www.dicomlibrary.com/.

[29] 2024 Medimodel Official Website. Last accessed on March 20. URL: https://medimodel.com/.

[30] WD Bidgood Jr and Steven C Horii. "Introduction to the ACR-NEMA DICOM standard." In: *Radiographics* 12.2 (1992), pp. 345–355.

[31] National Electrical Manufacturers Association et al. *NEMA PS3/ISO 12052, Digital Imaging and Communications in Medicine (DICOM) Standard*. 1993.

[32] Hai K Huang. *PACS and imaging informatics: basic principles and applications*. John Wiley & Sons, 2011.

[33] Luis Ibanez et al. "The ITK software guide second edition updated for ITK version 2.4". In: *The Insight Software Consortium* 9 (2005), pp. 503–524.

[34] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit (4th ed.)* Kitware, 2006. ISBN: 978-1-930934-19-1.

[35] Ivo Wolf et al. "The medical imaging interaction toolkit". In: *Medical image analysis* 9.6 (2005), pp. 594–604.

[36] Hui Dong et al. “Medical image reconstruction based on ITK and VTK”. In: *2013 International Conference on Computer Sciences and Applications*. IEEE. 2013, pp. 642–645.

[37] Hui Dong et al. “Medical Image Reconstruction Based on ITK and VTK”. In: *2013 International Conference on Computer Sciences and Applications*. 2013, pp. 642–645. DOI: 10.1109/CSA.2013.155.

[38] Jianhao Tan et al. “Design of 3D Visualization System Based on VTK Utilizing Marching Cubes and Ray Casting Algorithm”. In: *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*. Vol. 02. 2016, pp. 192–197. DOI: 10.1109/IHMSC.2016.153.

[39] Manh Ha Tran, Hoang Minh Quang Vu, et al. “A research on 3D model construction from 2D DICOM”. In: *2016 international conference on advanced computing and applications (ACOMP)*. IEEE. 2016, pp. 158–163.

[40] Rosemary Honea et al. “Evaluation of commercial PC-based DICOM image viewer”. In: *Journal of Digital Imaging* 11 (1998), pp. 151–155.

[41] Steven C Horii. “DICOM image viewers: a survey”. In: *Medical Imaging 2003: PACS and Integrated Medical Information Systems: Design and Evaluation*. Vol. 5033. SPIE. 2003, pp. 251–259.

[42] Wei Liao, Thomas M Deserno, and Klaus Spitzer. “Evaluation of free non-diagnostic DICOM software tools”. In: *Medical Imaging 2008: PACS and Imaging Informatics*. Vol. 6919. SPIE. 2008, pp. 11–22.

[43] Daniel Haak, Charles-E Page, and Thomas M Deserno. “A survey of DICOM viewer software to integrate clinical research and medical imaging”. In: *Journal of digital imaging* 29 (2016), pp. 206–215.

[44] Thomas M Deserno et al. “Digital imaging and electronic data capture in multicenter clinical trials”. In: *MedInfo*. 2015, p. 930.

[45] Giuseppe Lo Presti et al. “Assessment of DICOM viewers capable of loading patient-specific 3D models obtained by different segmentation platforms in the operating room”. In: *Journal of digital imaging* 28 (2015), pp. 518–527.

[46] Andreas Brühschwein et al. “Free DICOM-viewers for veterinary medicine: survey and comparison of functionality and user-friendliness of medical imaging PACS-DICOM-viewer freeware for specific use in veterinary medicine practices”. In: *Journal of digital imaging* 33 (2020), pp. 54–63.

[47] Jagjot Singh Wadali et al. “Evaluation of free, open-source, web-based DICOM viewers for the Indian national telemedicine service (eSanjeevani)”. In: *Journal of Digital Imaging* 33 (2020), pp. 1499–1513.

[48] Rúben Cruz. *Identisoft - Techincal document provided for the preparation of "Visualization and manipulation of medical images in modern web browsers.* 30-11-2022.

[49] *Medimodel Official*. 2022. URL: `https://medimodel.com/` (visited on 03/06/2024).

[50] Simon Thorpe, Denis Fize, and Catherine Marlot. “Speed of processing in the human visual system”. In: *nature* 381.6582 (1996), pp. 520–522.

Appendices

Appendix A

Published Articles

A.1 © CISTI'2024 – 19th Iberian Conference on Information Systems and Technologies

MIPRV – Medical Imagery Processing, Reconstruction and Visualization: A Web Platform Approach

Hugo Pereira
Instituto Politécnico de Viana do Castelo
Viana do Castelo, Portugal
hup@ipvc.pt

Pedro Miguel Faria
Instituto Politécnico de Viana do Castelo
Viana do Castelo, Portugal
pfaria@estg.ipvc.pt

Luís Romero
Instituto Politécnico de Viana do Castelo
Viana do Castelo, Portugal
romero@estg.ipvc.pt

Abstract — **Medical diagnosis has undergone a profound transformation, transitioning from traditional reliance on clinical intuition to modern methodologies incorporating comprehensive body scans for precise assessment. This evolution continues its shift in the analysis of medical scans from traditional 2D representations to advanced 3D reconstructions facilitated by computer software, recently through interactive medical imaging web applications. The article encompasses a thorough exploration of existing literature on DICOM files, similar solutions, and the capabilities of Cornerstone3D technology including a comparison of Cornerstone3D with its predecessor and a standalone DICOM viewing platform named MITK. With the aim of speeding up the diagnostic process, thus providing healthcare professionals with advanced tools for exhaustive analysis and informed decision-making, a web platform for visualizing DICOM images is proposed, to allow users to upload 2D scans in DICOM format and obtain their 3D reconstructions online, eliminating dependence on stand-alone applications, but maintaining the performance expected of this type of application.**

Keywords - DICOM Images, Cornerstone, Web Platform, Medical Images Visualization.

I. Introduction

In a hospital, during the patient diagnosis process, a scan of the patient's body or a crucial body part is often taken using X-ray technology. The resulting 2D print is then analyzed by medical professionals. Before the advent of X-ray machines, conditions such as broken bones, tumors, or swallowed objects, required invasive procedures and were mostly diagnosed through physical examination and a doctor's best guess [1]. With continuous innovations and technological advancements, the quality and ease of patient diagnosis have significantly improved. However, in the current era, medical professionals face challenges in acquiring critical information when viewing medical images in a 2D format. A 3D counterpart would enable them to navigate through a person's body scan more efficiently. Efforts have been made to develop software capable of processing medical images and reconstructing them into 3D visualizations. Identisoft, an enterprise specializing in information and communication technology solutions for the health sector, uses its expertise in medical imaging data management to innovate. One such solution is a Web platform designed to visualize 2D medical images in the DICOM (Digital Imaging and Communications in Medicine) format, which is a standard that plays a pivotal role in facilitating the seamless exchange and interoperability of medical images and related information [2]. This enterprise collaborates with the Polytechnic Institute of Viana do Castelo to introduce the development of a Web platform for medical image visualization, supported in web browsers. The objective of this Web platform is to incorporate tools like volume rendering and 3D reconstruction, using the Cornerstone3D library, a lightweight JavaScript library for visualizing medical images in modern web browsers. Cornerstone3D, along with its associated libraries, like Cornerstone3DTools, which offers a wide range of imaging tasks [3]. Numerous software solutions for reconstructing and visualizing medical images have been implemented, but this proposal takes a unique approach. It employs newer software (Cornerstone3D) and focuses on developing a web platform rather than a standalone application. The objective is to create a solution that leverages newer software within a web platform, aiming for enhancements outlined in the objectives section below. The subsequent section (Section 2) includes an extensive exploration of research and studies on DICOM files, similar work, and solutions in the field, along with a review of other published works on Cornerstone3D. Section 3 delves into the Cornerstone3D library, examining its uses, including image loading, volume loading, volume rendering, and other functions. Section 4 details the initial development and planning for the web platform, while Section 5 discusses some of the obtained results and Section 6 holds the conclusion.

II. Literature Review

This section reviews previous approaches, solutions, and technologies related to the processing and reconstruction of medical images from a 2D format to a 3D one. The research involves multiple published articles in this area to determine similar solutions to the proposal and to identify the work that has

Figure 17: Article published at the CISTI'2024 conference.

A.2 Journal of Digital Imaging - Springer

Recently changed to "Journal of Imaging Informatics in Medicine"

URL: https://link.springer.com/journal/10278

Web-Based DICOM Viewers: A Survey and a Performance Classification

Abstract

The standard for managing image data in healthcare is the DICOM (Digital Imaging and Communications in Medicine) protocol. DICOM web-viewers provide flexible and accessible platforms for their users to view and analyze DICOM images remotely. This article presents a comprehensive evaluation of various web-based DICOM viewers, emphasizing their performance in different rendering scenarios, browsers, and operating systems. The study includes a total of 16 web-based viewers, of which 12 were surveyed, and 7 were compared performance-wise based on the availability of an online demo. The criteria for examination include accessibility features, such as available information or requirements for usage, interface features, such as loading capabilities or cloud storage, two-dimensional (2D) viewing features, such as the ability to perform measurements or alter the viewing window, and three-dimensional (3D) viewing features, such as volume rendering or secondary reconstruction. Only 4 of the viewers allow for the viewing of local DICOM files in 3D (other than MPR(Multiplanar reconstruction)). Premium software offers a large amount of features with overall good performance. One of the free alternatives demonstrated the best efficiency in both 2D and 3D rendering but faces challenges with missing 3D rendering features in its interface, which is still in development. Other free options exhibited slower performance, especially in 2D rendering but have more ready-to-use features on their web app. The evaluation also underscores the importance of browser choice, with some browsers performing much better than the competition, and highlights the significance of hardware when dealing with rendering tasks.

Keywords: DICOM, Web-Viewer, Med3WEb, OHIF, PostDICOM, 3D, 2D, MEDDream

1 Introduction

In the fast-evolving landscape of modern medicine, the efficient management and interpretation of medical images are paramount to patient care [1]. Digital Imaging and Communications in Medicine, commonly known as DICOM [2], has emerged as the universal language of medical imaging, revolutionizing the way healthcare professionals access and analyze diagnostic images. DICOM was developed to address the challenges of interoperability in medical imaging, ensuring that images and associated patient

1

Figure 18: Article submitted for evaluation to the Journal of Digital Imaging.

Appendix B

MIPVR Platform

B.1 Mockups

Figure 19: Mockup of the Landing Page of The Platform.

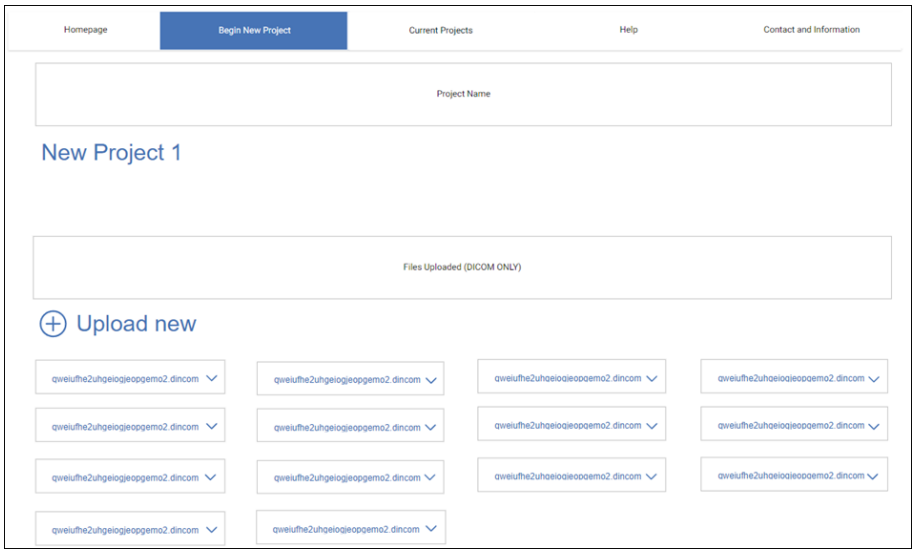


Figure 20: Mockup of the Uploading Page of The Platform.

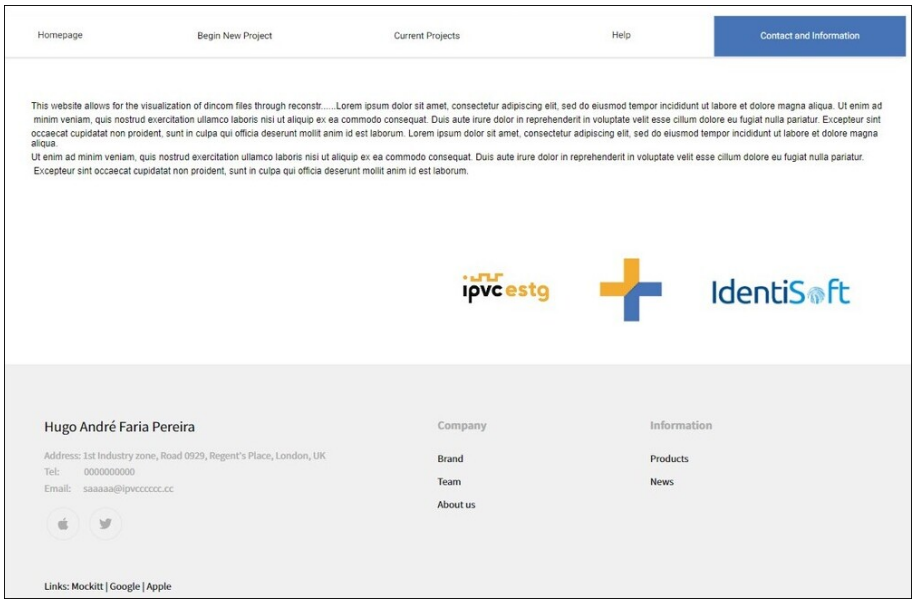


Figure 21: Mockup of the Help Page of The Platform.

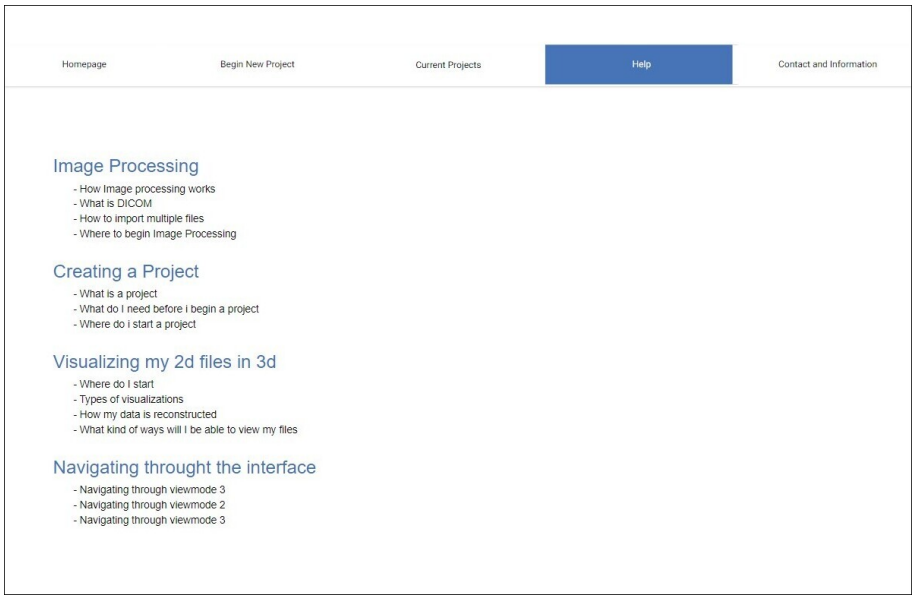
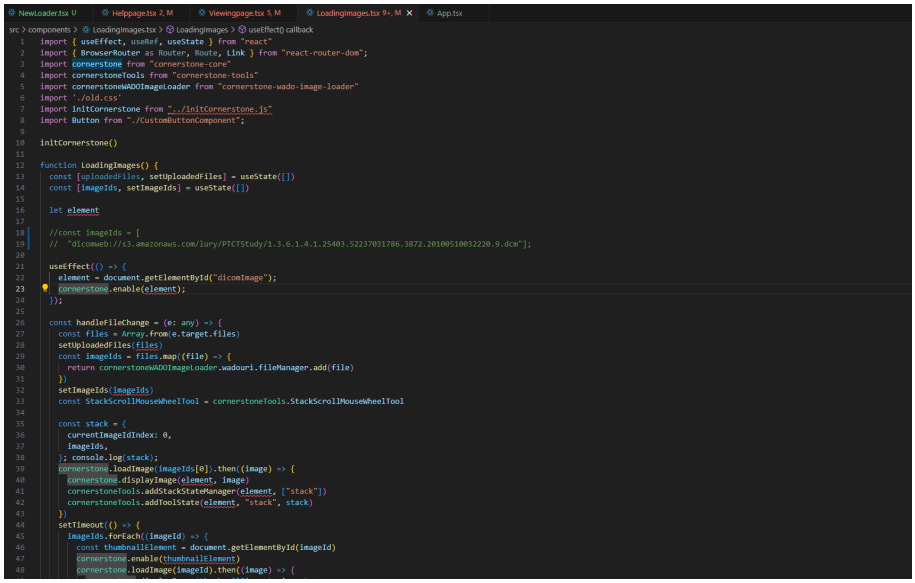


Figure 22: Mockup of the Contacts Page The Platform.

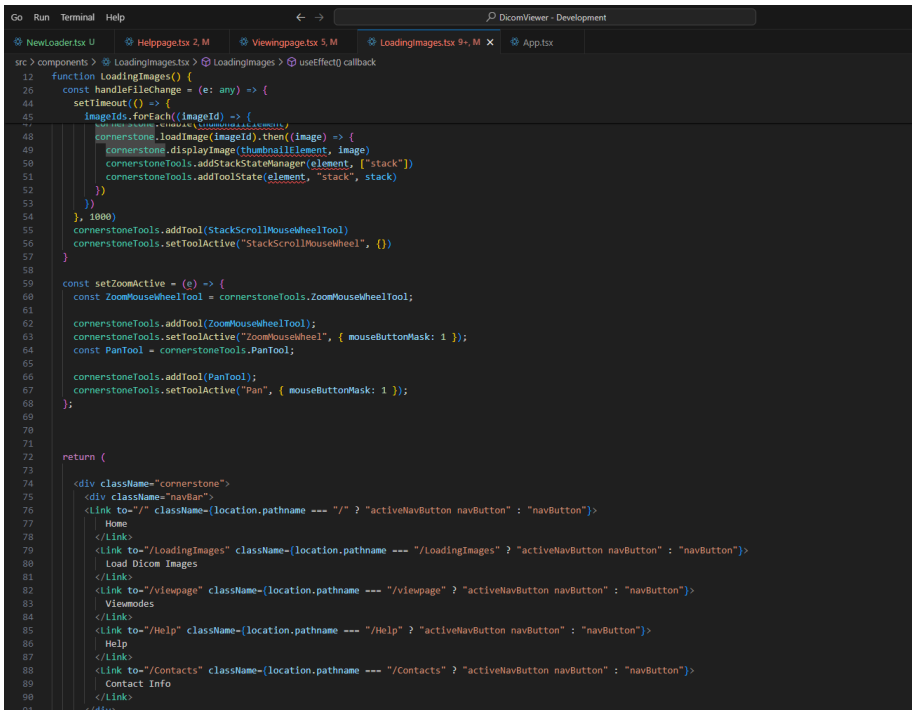
B.2 Loading Code

This section contains the code used to load files and view them while they are being loaded.



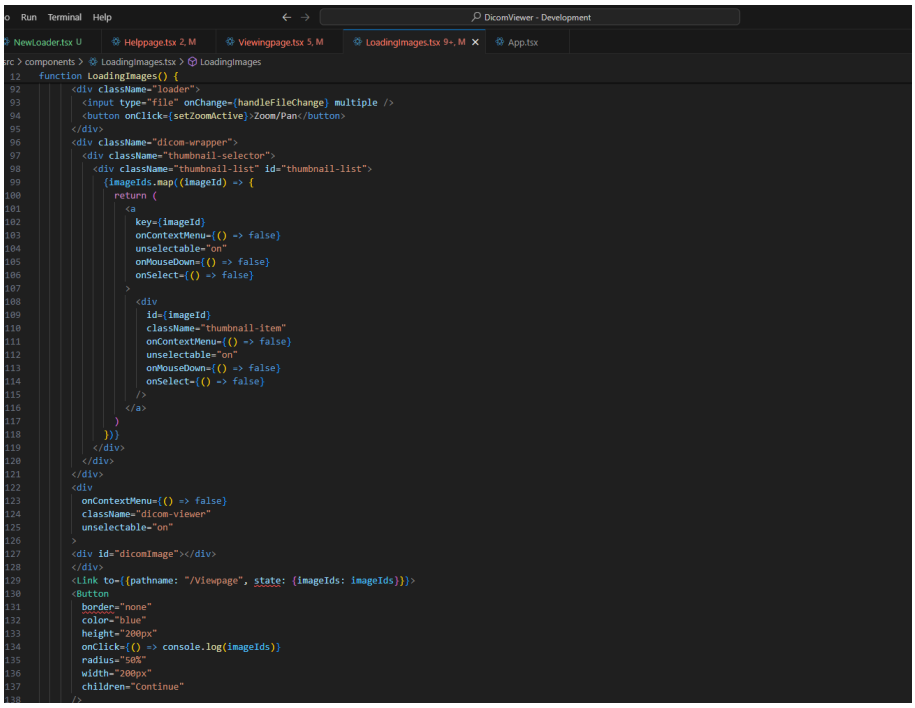
```
import { useEffect, useRef, useState } from "react"
import { BrowserRouter as Router, Route, Link } from "react-router-dom";
import cornerstone from "cornerstone-core"
import cornerstoneTools from "cornerstone-tools"
import cornerstoneWADOImageLoader from "cornerstone-wado-image-loader"
import './old.css'
import initCornerstone from "../initCornerstone.js"
import Button from "./CustomButtonComponent";

initCornerstone()

function LoadingImages() {
  const [uploadedFiles, setUploadedFiles] = useState([])
  const [imageIds, setImageIds] = useState([])

  let element

  //const imageIds = [
  // "dicomweb://s3.amazonaws.com/lury/PTCTStudy/1.3.6.1.4.1.25403.52237031786.3872.20100510032220.9.dcm"];

  useEffect(() => {
    element = document.getElementById("dicomImage");
    cornerstone.enable(element);
  });

  const handleFileChange = (e: any) => {
    const files = Array.from(e.target.files)
    setUploadedFiles(files)
    const imageIds = files.map((file) => {
      return cornerstoneWADOImageLoader.wadouri.fileManager.add(file)
    })
    setImageIds(imageIds)
    const StackScrollMouseWheelTool = cornerstoneTools.StackScrollMouseWheelTool

    const stack = {
      currentImageIdIndex: 0,
      imageIds,
    }; console.log(stack);
    cornerstone.loadImage(imageIds[0]).then((image) => {
      cornerstone.displayImage(element, image)
      cornerstoneTools.addStackStateManager(element, ["stack"])
      cornerstoneTools.addToolState(element, "stack", stack)
    })
    setTimeout(() => {
      imageIds.forEach((imageId) => {
        const thumbnailElement = document.getElementById(imageId)
        cornerstone.enable(thumbnailElement)
        cornerstone.loadImage(imageId).then((image) => {
```

Figure 23: Uploading Page Of The Web Platform Part 1.



```
function LoadingImages() {
  const handleFileChange = (e: any) => {
    setTimeout(() => {
      imageIds.forEach((imageId) => {
        cornerstone.loadImage(imageId).then((image) => {
          cornerstone.displayImage(thumbnailElement, image)
          cornerstoneTools.addStackStateManager(element, ["stack"])
          cornerstoneTools.addToolState(element, "stack", stack)
        })
      })
    }, 1000)
    cornerstoneTools.addTool(StackScrollMouseWheelTool)
    cornerstoneTools.setToolActive("StackScrollMouseWheel", {})
  }

  const setZoomActive = (e) => {
    const ZoomMouseWheelTool = cornerstoneTools.ZoomMouseWheelTool;

    cornerstoneTools.addTool(ZoomMouseWheelTool);
    cornerstoneTools.setToolActive("ZoomMouseWheel", { mouseButtonMask: 1 });
    const PanTool = cornerstoneTools.PanTool;

    cornerstoneTools.addTool(PanTool);
    cornerstoneTools.setToolActive("Pan", { mouseButtonMask: 1 });
  };



  return (

    <div className="cornerstone">
      <div className="navBar">
      <Link to="/" className={location.pathname === "/" ? "activeNavButton navButton" : "navButton"}>
          Home
        </Link>
        <Link to="/LoadingImages" className={location.pathname === "/LoadingImages" ? "activeNavButton navButton" : "navButton"}>
          Load Dicom Images
        </Link>
        <Link to="/viewpage" className={location.pathname === "/viewpage" ? "activeNavButton navButton" : "navButton"}>
          Viewmodes
        </Link>
        <Link to="/Help" className={location.pathname === "/Help" ? "activeNavButton navButton" : "navButton"}>
          Help
        </Link>
        <Link to="/Contacts" className={location.pathname === "/Contacts" ? "activeNavButton navButton" : "navButton"}>
          Contact Info
        </Link>
```

Figure 24: Uploading Page Of The Web Platform Part 2.

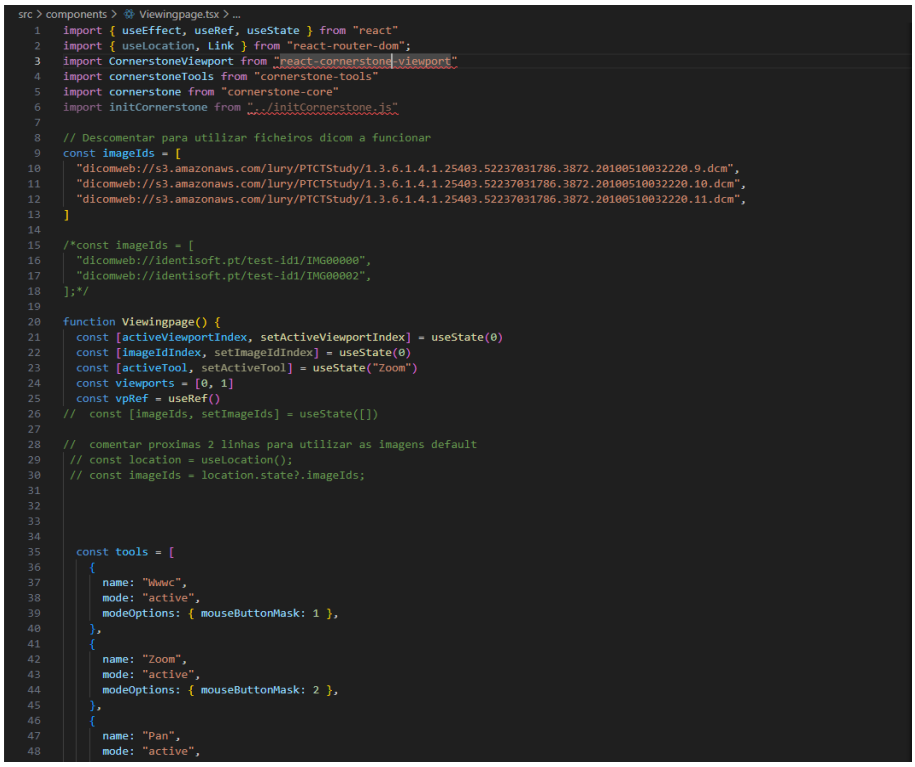


```
function LoadingImages() {
      <div className="loader">
        <input type="file" onChange={handleFileChange} multiple />
        <button onClick={setZoomActive}>Zoom/Pan</button>
      </div>
      <div className="dicom-wrapper">
        <div className="thumbnail-selector">
          <div className="thumbnail-list" id="thumbnail-list">
            {imageIds.map((imageId) => {
              return (
                <a
                  key={imageId}
                  onContextMenu={() => false}
                  unselectable="on"
                  onMouseDown={() => false}
                  onSelect={() => false}
                >
                  <div
                    id={imageId}
                    className="thumbnail-item"
                    onContextMenu={() => false}
                    unselectable="on"
                    onMouseDown={() => false}
                    onSelect={() => false}
                  />
                </a>
              )
            })}
          </div>
        </div>
      </div>
      <div
        onContextMenu={() => false}
        className="dicom-viewer"
        unselectable="on"
      >
      <div id="dicomImage"></div>
      </div>
      <Link to={{pathname: "/Viewpage", state: {imageIds: imageIds}}}>
      <Button
        border="none"
        color="blue"
        height="200px"
        onClick={() => console.log(imageIds)}
        radius="50%"
        width="200px"
        children="Continue"
      />
```

Figure 25: Uploading Page Of The Web Platform Part 3.

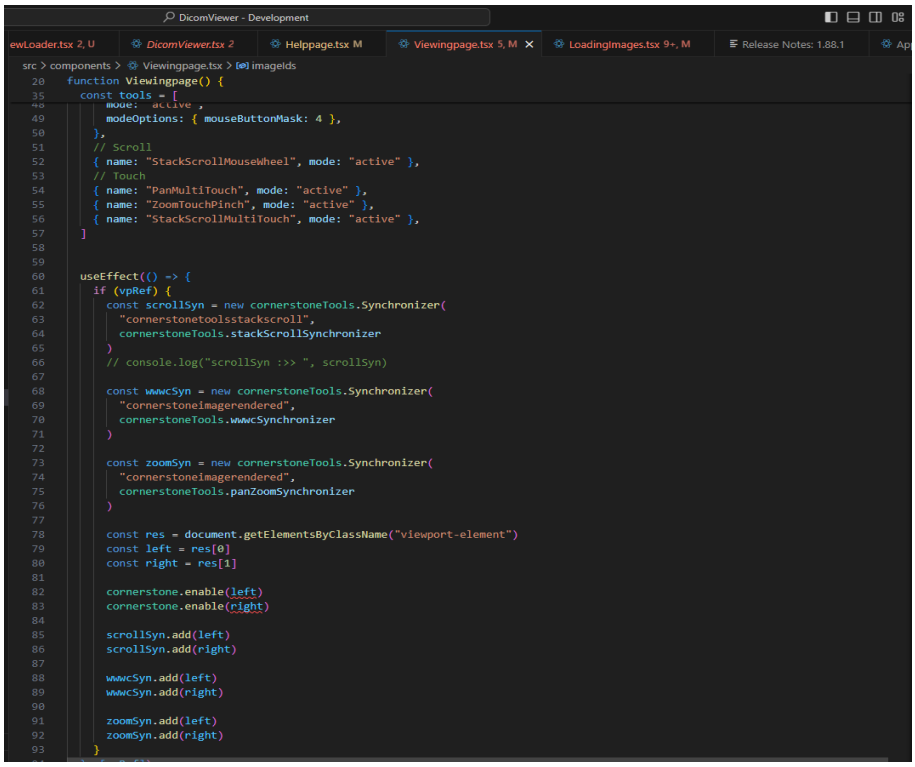
B.3 Viewing Code

This section contains the code to view files with Cornerstone3D viewports.



```
src > components > Viewingpage.tsx > ...
import { useEffect, useRef, useState } from "react"
import { useLocation, Link } from "react-router-dom";
import CornerstoneViewport from "react-cornerstone-viewport"
import cornerstoneTools from "cornerstone-tools"
import cornerstone from "cornerstone-core"
import initCornerstone from "../initCornerstone.js"

// Descomentar para utilizar ficheiros dicom a funcionar
const imageIds = [
  "dicomweb://s3.amazonaws.com/lury/PTCTStudy/1.3.6.1.4.1.25403.52237031786.3872.20100510032220.9.dcm",
  "dicomweb://s3.amazonaws.com/lury/PTCTStudy/1.3.6.1.4.1.25403.52237031786.3872.20100510032220.10.dcm",
  "dicomweb://s3.amazonaws.com/lury/PTCTStudy/1.3.6.1.4.1.25403.52237031786.3872.20100510032220.11.dcm",
]

/*const imageIds = [
  "dicomweb://identisoft.pt/test-id1/IMG00000",
  "dicomweb://identisoft.pt/test-id1/IMG00002",
];*/

function Viewingpage() {
  const [activeViewportIndex, setActiveViewportIndex] = useState(0)
  const [imageIdIndex, setImageIdIndex] = useState(0)
  const [activeTool, setActiveTool] = useState("Zoom")
  const viewports = [0, 1]
  const vpRef = useRef()
//  const [imageIds, setImageIds] = useState([])

//  comentar proximas 2 linhas para utilizar as imagens default
 // const location = useLocation();
 // const imageIds = location.state?.imageIds;



  const tools = [
    {
      name: "Wwwc",
      mode: "active",
      modeOptions: { mouseButtonMask: 1 },
    },
    {
      name: "Zoom",
      mode: "active",
      modeOptions: { mouseButtonMask: 2 },
    },
    {
      name: "Pan",
      mode: "active",
```

Figure 26: Viewing Page Of The Web Platform Part 1.



```
DicomViewer - Development
ewLoader.tsx 2, U | DicomViewer.tsx 2 | Helppage.tsx M | Viewingpage.tsx 5, M | LoadingImages.tsx 9+, M | Release Notes: 1.88.1
src > components > Viewingpage.tsx > imageIds
function Viewingpage() {
  const tools = [
      mode: "active",
      modeOptions: { mouseButtonMask: 4 },
    },
    // Scroll
    { name: "StackScrollMouseWheel", mode: "active" },
    // Touch
    { name: "PanMultiTouch", mode: "active" },
    { name: "ZoomTouchPinch", mode: "active" },
    { name: "StackScrollMultiTouch", mode: "active" },
  ]


  useEffect(() => {
    if (vpRef) {
      const scrollSyn = new cornerstoneTools.Synchronizer(
        "cornerstonetoolsstackscroll",
        cornerstoneTools.stackScrollSynchronizer
      )
      // console.log("scrollSyn :>> ", scrollSyn)

      const wwwcSyn = new cornerstoneTools.Synchronizer(
        "cornerstoneimagerendered",
        cornerstoneTools.wwwcSynchronizer
      )

      const zoomSyn = new cornerstoneTools.Synchronizer(
        "cornerstoneimagerendered",
        cornerstoneTools.panZoomSynchronizer
      )

      const res = document.getElementsByClassName("viewport-element")
      const left = res[0]
      const right = res[1]

      cornerstone.enable(left)
      cornerstone.enable(right)

      scrollSyn.add(left)
      scrollSyn.add(right)

      wwwcSyn.add(left)
      wwwcSyn.add(right)

      zoomSyn.add(left)
      zoomSyn.add(right)
    }
```

Figure 27: Viewing Page Of The Web Platform Part 2.

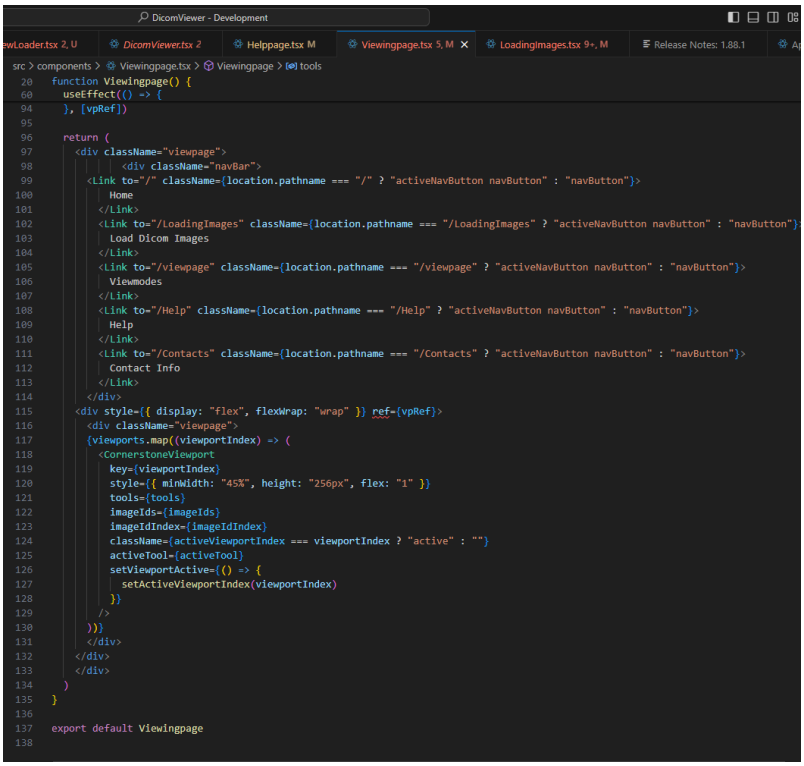


Figure 28: Viewing Page Of The Web Platform Part 3.

Appendix C

Computer specifications

- **Computer A:**
  1. Processor : AMD Ryzen 5 5600X 6-Core Processor
  2. Video Card : NVIDIA GeForce RTX 3080
  3. Operating System : Windows 11
  4. RAM : 16 GB
- **Computer B:**
  1. Processor : AMD Ryzen 5 5600H with Radeon Graphics
  2. Video Card : NVIDIA GeForce RTX 3060 Laptop GPU
  3. Operating System : Windows 11
  4. RAM : 16 GB
- **Computer C:**
  1. Processor : Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
  2. Video Card : NVIDIA GeForce GTX 1060
  3. Video Card #2 : Intel(R) HD Graphics 630
  4. Operating System : Windows 10
  5. RAM : 16 GB
- **Computer D:**

1. Processor : 4,2 GHz Intel Core 7
2. Video Card : Radeon Pro 580 8192 MB
3. Operating System : macOS High Sierra version 10.13.6
4. RAM : 16 GB 2400 MHz DDR4

- **Computer E:**

1. Processor : 3 GHz 8-Core Intel Xeon E5
2. Video Card : AMD firePro D700 6 GB
3. Operating System : macOS Monterey version 12.6.3
4. RAM : 16 GB 1866 MHz DDR3

Appendix D

Time Breakdowns

D.1 OHIF Time breakdowns

Table 11: OHIF Test 1 Times.

Browser	Computer	Breakdown (s)
Firefox	A	(00.25 + 00.93)
	B	(00.29 + 00.64)
	C	(00.49 + 01.50)
Edge	A	(00.56 + 00.80)
	B	(00.81 + 00.66)
	C	(01.58 + 01.03)
Chrome	A	(00.31 + 00.49)
	B	(00.63 + 00.61)
	C	(01.26 + 01.30)
	D	(01.18 + 01.20)
	E	(01.27 + 01.34)

Table 12: OHIF Test 2 Times.

Browser	Computer	Breakdown (s)
Firefox	A	(00.20 + 00.64)
	B	(00.28 + 00.09)
	C	(00.64 + 00.34)
Edge	A	(00.62 + 00.85)
	B	(00.68 + 00.49)
	C	(01.52 + 00.63)
Chrome	A	(00.34 + 00.38)
	B	(00.49 + 00.15)
	C	(00.99 + 00.30)
	D	(01.07 + 0.24)
	E	(01.01 + 00.61)

D.2 MED3WEB Time breakdowns

Table 13: MED3WEB Test 1 Times.

Browser	Computer	Breakdown (s)
Firefox	A	(07.56 + 01.76)
	B	(07.76 + 02.13)
	C	(24.13 + 04.13)
Edge	A	(90.54 + 61.65)
	B	(93.22 + 61.97)
	C	(179.85 + 129.29)
Chrome	A	(04.99 + 02.43)
	B	(05.50 + 02.72)
	C	(15.77 + 05.63)
	D	(10.48 + 05.08)
	E	(10.52 + 05.35)

Table 14: MED3WEB Test 2 Times.

Browser	Computer	Breakdown (s)
Firefox	A	(09.04 + 01.49)
	B	(05.68 + 01.74)
	C	(19.46 + 04.36)
Edge	A	(90.75 + 61.22)
	B	(93.40 + 62.62)
	C	(180.96 + 121.66)
Chrome	A	(04.91 + 02.49)
	B	(12.29 + 02.70)
	C	(16.15 + 05.05)
	D	(10.57 + 05.02)
	E	(10.37 + 05.59)