Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Engenharia Informática

Dissertação

Innovative Solutions for Healthcare Data Management: The Medical Data Hub Approach

Jose Hugo Sousa Silva

Orientador(es) | Teresa Gonçalves

Miguel Angel Guevara Lopez

Évora 2024

Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Engenharia Informática

Dissertação

Innovative Solutions for Healthcare Data Management: The Medical Data Hub Approach

Jose Hugo Sousa Silva

Orientador(es) | Teresa Gonçalves

Miguel Angel Guevara Lopez

Évora 2024

A dissertação foi objeto de apreciação e discussão pública pelo seguinte júri nomeado pelo Diretor da Escola de Ciências e Tecnologia:

Presidente	|	José Saias (Universidade de Évora)
Vogais	|	Pedro Salgueiro (Universidade de Évora) Teresa Gonçalves (Universidade de Évora) (Orientador)

Évora 2024

Declaration

I affirm that this Dissertation is entirely my original work and has not been submitted for a degree at this or any other institution.

I have reviewed and comprehend the plagiarism guidelines outlined in the General Regulations of the University Calendar for the current year, available at `https://gdoc.uevora.pt/400756`.

Signed:  Date: 11/11/2024

Abstract

The introduction of new data privacy regulations poses a significant challenge for medical research, particularly in accessing and utilizing medical data outside the hospital environment.

The use of medical data is essential for developing new technologies, such as machine learning models, which can help save lives by identifying diseases like cancer at early stages. Machine learning requires large amounts of data to train models, and the lack of data presents a bottleneck for the development of new, more robust and precise models that can predict diseases and save human lives.

This document describes a platform designed to store medical data securely and anonymously, while complying with the new regulatory environment aiming to provide researchers and medical professionals with a tool that allows them to access and upload medical data, and utilize it for training machine learning models.

The developed system features a web-based graphical interface that allows researchers and medical professionals to access compliant medical data. Users can log in, sign up, create patient and study records with unique identifiers, attach studies to patients, read DICOM (including digitized film mammography images) files, anonymize the images within these files and explore the images.

The platform was designed with the input of medical professionals and researchers, and it is intended to be used in a real-world setting. By addressing the data accessibility and privacy challenges, this platform has the potential to significantly advance medical research and the development of life-saving technologies.

Sumário

A introdução de novas regulamentações de privacidade de dados representa um desafio significativo para a investigação médica, particularmente no acesso e utilização de dados médicos fora do ambiente hospitalar.

A utilização de dados médicos é essencial para o desenvolvimento de novas tecnologias, como modelos de aprendizagem automática, que podem ajudar a salvar vidas ao identificar doenças como o cancro em estádios iniciais. A aprendizagem automática requer grandes quantidades de dados para treinar modelos, e a falta de dados apresenta um obstáculo para o desenvolvimento de novos modelos mais robustos e precisos que possam identificar doenças e salvar vidas humanas.

Este documento descreve uma plataforma projectada para armazenar dados médicos de forma segura e anónima, enquanto cumpre com o novo ambiente regulatório. A plataforma visa fornecer aos investigadores e profissionais de saúde uma ferramenta que lhes permita aceder e carregar dados médicos, e utilizá-los para treinar modelos de aprendizagem automática.

O sistema desenvolvido possui uma interface gráfica baseada na web que permite aos investigadores e profissionais de saúde aceder a dados médicos em conformidade com as regulamentações. Os utilizadores podem efectuar login, inscrever-se, criar registos de pacientes e estudos com identificadores únicos, associar estudos a pacientes, ler ficheiros DICOM (incluindo imagens de mamografia digitalizadas), anonimizar as imagens nesses ficheiros e explorar as imagens.

A plataforma foi concebida com a participação de profissionais de saúde e investigadores, e é destinada a ser utilizada num cenário real. Ao abordar os desafios de acessibilidade e privacidade dos dados, esta plataforma tem o potencial de avançar significativamente a investigação médica e o desenvolvimento de tecnologias que salvem vidas.

Acknowledgements

I would like to provide my deepest gratitude to my supervisors, Teresa Gonçalves and Miguel Lopez, for their invaluable guidance and trust deposited in me throughout the project. Their expertise and advice have been instrumental in the successful completion of this project, and I am truly grateful for their help.

I would also like to thank my family and friends for their trustworthiness and support over the past few months. Your belief in me has been a constant source of motivation.

Contents

List of Figures

List of Tables

Nomenclature

ACID	Atomicity, Consistency, Isolation, Durability
ACR	American College of Radiology
API	Application Programming Interface
CD	Continuous Development
CI	Continuous Integration
CORS	Cross-Origin Resource Sharing
CRUD	Create, Read, Update, Delete
CSV	Comma-separated values
CT	Computed Tomography
DICOM	Digital Imaging and Communications in Medicine
DOM	Document Object Model
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment
IODs	Information Object Definitions
IP	Internet Protocol
LDAP	Lightweight Directory Access Protocol
MRI	Magnetic Resonance Imaging
MVP	Minimum Viable Product
NEMA	National Electrical Manufacturers Association
OSI	Open Systems Interconnection
PaaS	Platform as a Service Product
RBAC	Role-Based Access Control
SEO	Search Engine Optimization
SPA	Single Page Application
SQL	Structured Query Language
SSG	Static Site Generator
SSR	Server-Side Rendering
TCP	Transmission Control Protocol

1 Introduction

The present chapter describes the motivation behind this work, emphasizing the critical need for early disease detection and the limitations faced currently in the medical research field in Europe. It introduces the context regarding the collaborative environment in which the project was developed and outlines the primary objectives. Finally, it provides a summary of the document's structure detailing each chapter.

1.1 Motivation

Cancer remains a critical global health issue, with an estimated 22 million new cases and 9.7 million deaths reported in 2022; the most common types of cancer include lung, breast, colon, rectum and prostate cancer; female breast cancer, in particular is the second most common cancer in the world, with an estimated of 2,23 million new cases in 2022 [3]. Early detection of diseases like breast cancer is vital for effective treatment and improving survival rates; the earlier a disease is detected, the better the chances of successful treatment and survival [4].

Despite the importance of early detection, it's estimated that 1.3% to 35.9% of cancers detected clinically between mammography screening examinations and screen-detected cancers can be classified as missed diagnoses (weren't correctly identified as positive) [5]. This underscores the need for better detection methods.

Machine Learning models have shown highly positive results in the detection of breast cancer, with some studies showing an accuracy of up to 98% [6, 7]. However, despite advances in medical technology, Europe currently lacks a centralized platform for storing and managing medical data outside hospitals and clinics [8, 9, 10]. This creates a problem for researchers and medical professionals, who need access to large amounts of data in order to research and develop new treatments, as well as for the detection of diseases such as breast cancer.

Without centralized data storage, researchers don't possess enough data to investigate and to train machine learning models that can predict diseases. This lack of data prevents the development of new technologies that could be helpful to detect diseases and ultimately save human lives.

1.2 Context

This research and development project was carried out in collaboration with the Setúbal Polytechnic University and the Breast Unit of the Local Health Unit of Arrábida, Setúbal. The partnership between these institutions played a crucial role in the successful execution of this work, mainly in the definition of the data model to capture the appropriate features of patient cases, which allowed to significantly improve the quality and relevance of this project. With this, we ensure that the platform is able to store the crucial patient case data/information needed to support both the scientific and practical challenges in the field of medical data management.

1.3 Objectives

The main objective of this work is to develop a centralized platform, the Medical Data Hub, designed to securely store, manage, and analyze medical data. In this sense, to test and validate the platform we will focus on cancer data, and as a use case we have selected breast cancer. The platform will also support the study of other diseases, such as lung cancer, brain cancer, heart diseases, and other conditions that can be detected through medical imaging, blood tests, and other diagnostic methods, the primary focus will be on breast cancer, given its high prevalence and impact on global health.

The platform aims to address the current challenges of data scarcity and compliance with data protection regulations, enabling researchers and medical professionals to access and utilize large datasets for medical research and the development of new diagnostic tools.

1.4 Approach

To achieve these challenges and objective, the Medical Data Hub platform will be designed with the following key features:

- **Secure Data Storage:** Ensuring compliance with GDPR regulations through secure data storage, encryption, and anonymization techniques.
- **Dynamic Data Insertion:** Allowing users to dynamically insert various types of data, including DICOM images and patient information.
- **Data Visualization:** Enabling users to visualize medical images and other data within the platform.
- **Data Export:** Providing mechanisms for exporting data for use in machine learning models and other analytical tools.
- **Machine Learning Integration:** Allowing users to apply machine learning models directly within the platform.

The Medical Data Hub platform was designed with the capability to integrate machine learning models and data export features. However, the implementation of these features were not objectives of this work, but planned for future development.

1.5 Contributions

The present work aims to contribute to both the field of medical research and software engineering.

Contributions to medical research include:

- Providing a centralized, secure platform for storing and managing anonymized medical data.
- Facilitating access to large datasets for researchers and medical professionals.
- Supporting the development of machine learning models for early disease detection.
- Ensuring compliance with European data protection regulations, thereby protecting patient privacy and data security.

Contributions to software engineering include:

- Designing a scalable system architecture for secure and efficient medical data management, supporting large volumes of diverse data types, such as medical images and pa-

tient information.

- Leveraging microservices architecture and containerization to enhance modularity, deployment ease, and maintainability, contributing to best practices in scalable software design.

1.6 Structure of the Document

The present document is structured as follows:

- **Chapter 2: The Problem** – Discusses the challenges associated with current medical data storage practices, the lack of accessible data, and regulatory issues.
- **Chapter 3: State of the Art** – Reviews the current state of the art in medical data storage.
- **Chapter 4: Solution Proposal** – Describes the proposed Medical Data Hub platform, its features, requirements, design, implementation, support and security.
- **Chapter 5: Technologies** – Discusses the technologies used in the development of the platform, including Python, Pydicom, Presidio Image Redactor, Next.js, and Docker.
- **Chapter 6: Results and Discussion** – Presents the Medical Data Hub Platform, discussing its features and limitations.
- **Chapter 7: Conclusion and Future Work** – Summarizes the results, discusses the impact of the platform, and suggests directions for future work.

2 | Problem

The field of medical research faces several significant challenges, particularly in accessing and utilizing medical data outside the hospital environment.

This chapter outlines two primary issues that create bottlenecks to the advancement of medical research: the lack of freely available data and the complex regulatory landscape governing medical data in Europe.

2.1 Lack of freely available Data

Researchers and medical professionals require access to large amounts of data in order to develop and study diseases. Currently, medical data is primarily stored within hospitals and clinics, due to the sensitive nature of the data, the difficulty in extracting the data and the need to comply with data protection regulations. The difficulty in extracting data occurs due to the nature of the data and the systems present in the hospital environment that don't allow easily to extract the data, according to the conducted research. This due to the systems they use, or regulations on the hospital environment that prevent the data from being extracted.

Machine learning models require a large amount of data to be trained on, the bigger the dataset the better the model will perform [11]. So it's crucial to have bigger data sets to train this models, as they present a huge opportunity to save people life's, by detecting diseases earlier and more accurately.

2.2 Regulation

The regulation of medical data is a complex and evolving field, with the introduction of the General Data Protection Regulation (GDPR) in 2016 [12], all the entities that process personal

data must comply with the regulation by 2018. This includes the storage and processing of medical data.

Medical data lies in a special category of data [13], which requires additional safeguards to protect its privacy and security. This regulation presents a challenge as data must be stored and processed in a way that is compliant with the regulation.

The data must follow the GDPR, which include, but are not limited to, the following principles [14]:

- Data must be stored securely and encrypted.
- Data must be anonymized.
- Data must be accurate and up-to-date.
- Data must be minimized in relation to the purpose for which they are processed.

Understanding and implementing these regulations is crucial for creating a centralized medical data platform that is both effective and compliant with European laws.

3 State of the Art

The present chapter describes the current state of the art in the field of medical data storage, management, and analysis. It includes a review of the DICOM standard, existent solutions and a comparison of various platforms, and finally a discussion of their features and limitations.

3.1 DICOM Standard

DICOM stands for Digital Imaging and Communications in Medicine, and it's the standard utilized for the communication and management of medical imaging data and associated information; this standard was developed by the National Electrical Manufacturers Association (NEMA) and the American College of Radiology (ACR) in 1983, and it's currently maintained by the DICOM Standards Committee [15, 16].

As technology progressed, the amount of data generated by medical imaging devices increased, and the need for a standard format to store and exchange this data between different device manufacture became essential.

3.1.1 Structure and Format

The DICOM standard defines both a file format and a communication protocol. The file format is used to store medical imaging data, and the communication protocol is used to exchange this data between devices.

Information hierarchy

The DICOM information is structured hierarchically, where the root of the hierarchy is the patient, and the information is organized in a tree-like structure. The hierarchy is composed of the following levels: patient, study, series, and image.

A patient can have multiple studies, each study can have multiple series, and each series can have multiple images.

The order of the hierarchy is illustrated in the figure 3.1.

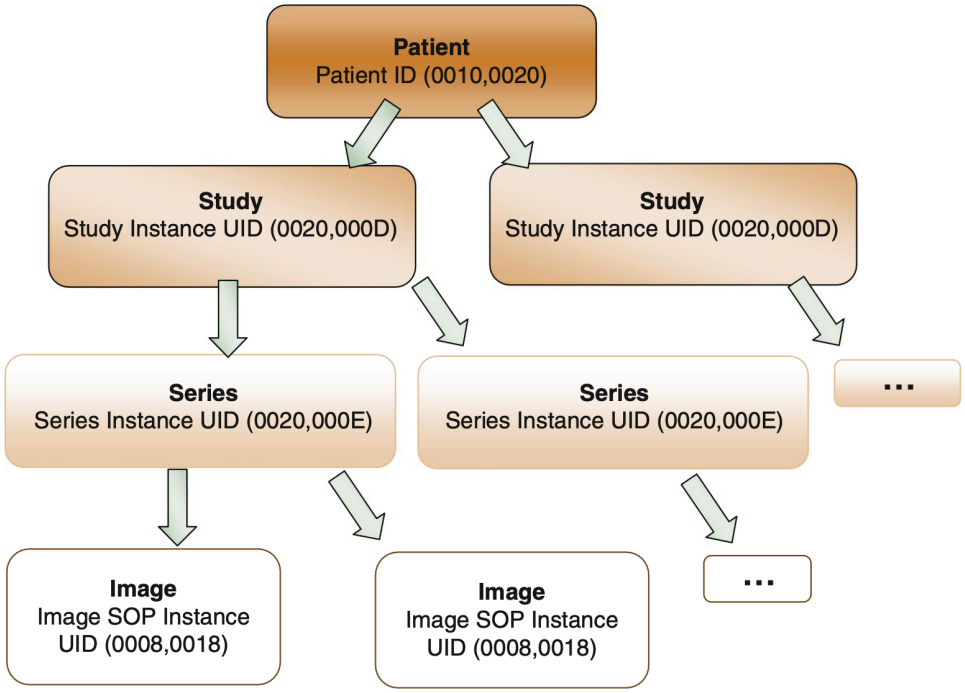

Figure 3.1: DICOM information hierarchy. Source: [1]

File Format

The file format of a DICOM files consists of a header and image data.

- Header: contains metadata information about the patient, study, image acquisition parameters, and other relevant data.
- Image Data: contains the pixel data of the image and its associated information.

Header. The metadata present in the header represents information about the real-world, and it's stored in a key-value format, where the key is the tag of the attribute and the value is the data itself. The information is stored in accordance with DICOM Information Object Definitions (IODs).

IODs are a collection of attributes that define the information that can be stored in a DICOM file. Each IOD is defined by a unique identifier, and it specifies the attributes that can be included in a DICOM file. For instance for a patient, the IOD would include attributes such as patient name, patient ID, date of birth, and others, as illustrated in the Figure 3.2.

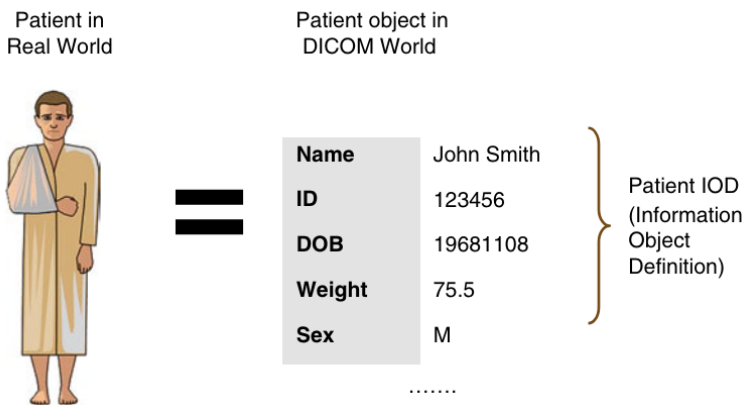

Figure 3.2: Example of patient information stored in a IOD. Source: [1]

Image Data. The image data is stored in a pixel data element, and it's represented in a binary format. It contains the actual size of the image and pixel data itself. In many cases the pixel data represents 95% of the DICOM file size [1].

The image data can encapsulate a sequence of images frames in a single file. This is useful for storing dynamic images such as ultrasound or MRI.

DICOMDIR

The DICOMDIR is a directory of DICOM files, it's used to organize and index DICOM files. It plays a similar role to the file system directory, creating a small database of DICOM files. Usually DICOM files are stored in a CD or DVD, with the DICOMDIR format, being a common way to distribute medical imaging data to patients. The DICOMDIR structure is illustrated in the figure 3.3.

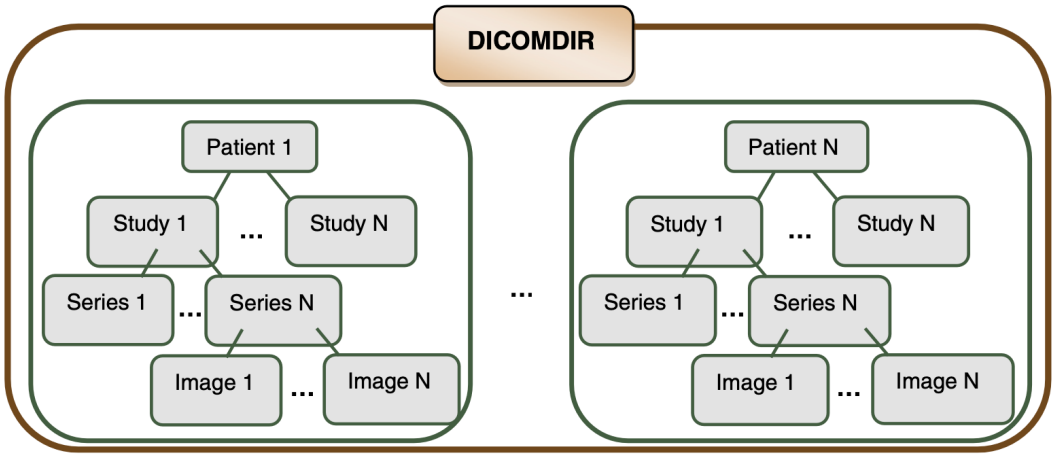

Figure 3.3: DICOMDIR structure. Each image represents a DICOM file. Source: [1]

Communication Protocol

The DICOM communication protocol is used to exchange medical imaging data between devices. It's based on the Open Systems Interconnection (OSI) model and uses the Transmission Control Protocol (TCP) and the Internet Protocol (IP) protocol for communication, allowing the transfer of images and related information between devices such as scanners, servers, workstations, and printers [17].

3.1.2 Importance

The DICOM standard is essential for ensuring that medical data can be stored and exchanged consistently and interoperability with different devices manufacturers and institutions. Additionally, the DICOM standard offers significant benefits, including:

- **Security**: DICOM includes features for ensuring the security and privacy of medical imaging data.
- **Efficiency**: DICOM allows for the efficient storage and retrieval of medical imaging data, reducing the time and effort required to access and process this data.
- **Comprehensive Information**: DICOM allows for the storage of comprehensive information about the patient, study, and image, ensuring that all relevant data is available for analysis and diagnosis.

3.1.3 Examples of DICOM Images

DICOM images are used in a wide range of medical imaging applications, including:

- **X-ray**: used for imaging bones and other dense tissues;
- **Computed Tomography (CT)**: used for imaging internal organs and tissues;
- **Magnetic Resonance Imaging (MRI)**: used for imaging soft tissues and organs;
- **Ultrasound**: used for imaging internal organs and tissues [18].

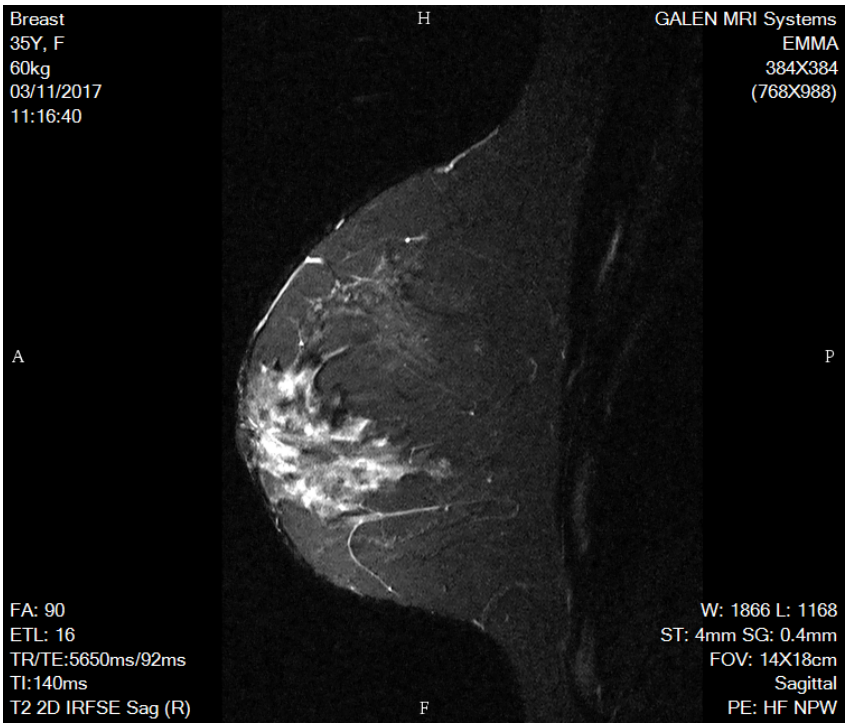

Figure 3.4: Example of a Breast MRI. Source: [2]

3.1.4 Conclusion

As discussed in this chapter, the DICOM standard is the standard used for the communication and management of medical imaging information and related data in the medical imaging field.

The standard is relevant as it ensures that medical imaging data can be stored and exchanged consistently and standardized.

The understanding of the DICOM standard is essential for anyone working with medical imaging data, as it provides the foundation for the storage and processing of this data. As such, the DICOM standard is vital for the development of our proposed platform.

3.2 Current Technologies and Platforms

The field of medical data storage, management, and analysis is rapidly evolving, with new technologies and platforms being developed to address the growing need for efficient and effective data management.

3.2.1 European Cancer Information System (ECIS)

The European Cancer Information System is a platform that provides access to a wide range of cancer data, including information on cancer incidence, mortality, and others [19].

Features

- Statistical data analysis tools.
- Information storages about cancer incidence, mortality, and other related data.
- Data visualization and aggregation tools.

Limitations

- Only provides statistical data.
- Lacks images, or any other relevant information for research.

3.2.2 DICOM Library

The DICOM Library is a free platform that provides access to a large collection of DICOM images and videos for educational and scientific purposes. Additionally, it provides the tools for uploading, downloading, and sharing DICOM files while anonymizing patient data [20].

Features

- Large collection of cancer imaging data.
- Data analysis and visualization tools.
- Data download and sharing capabilities.
- Data anonymization to protect patient privacy.
- Allows users to upload new data.

Limitations

- Does not allow application of machine learning models directly on the platform.
- Lacks tools for data preprocessing.

3.2.3 Cancer Imaging Archive (TCIA)

The Cancer Imaging Archive is a platform that provides access to a large collection of cancer imaging data [21].

Features

- Access to a large collection of cancer imaging data.
- Data analysis and visualization tools.
- Data download and sharing capabilities.
- Data anonymization to protect patient privacy.
- Allows users to upload new data.

Limitations

- Does not allow application of machine learning models directly on the platform.
- Lacks tools for data preprocessing.
- Located outside the European Union.

3.2.4 UK Biobank

The UK Biobank is a large-scale biomedical database that provides access to a wide range of biomedical data [22].

Features

- Access to a wide range of health-related data.
- Data anonymization to protect patient privacy.
- Data download and sharing capabilities.

Limitations

- Does not have cancer imaging data.
- Lacks tools for data analysis and visualization.
- Does not provide tools for data preprocessing.
- Users cannot upload new data.
- Does not support direct application of machine learning models.
- Located outside the European Union.

3.2.5 Genomic Data Commons Data Portal

The Genomic Data Commons Data Portal is a repository that provides access to a wide range of genomic data [23].

Features

- Access to a wide range of genomic data.
- Tools for data analysis and visualization.
- Data download and sharing capabilities.

Limitations

- Lacks tools for data preprocessing.
- Users cannot upload new data.
- Does not support direct application of machine learning models.
- Located outside the European Union.

3.2.6 Breast Cancer Digital Repository (BCDR)

The Breast Cancer Digital Repository is a platform that provides access to a collection of 1734 breast cancer image data and metadata [24].

Features

- Access to a collection of 1734 breast cancer image data and metadata.
- Tools for data analysis and visualization.
- Data download and sharing capabilities.

Limitations

- Lacks tools for data preprocessing.
- Users cannot upload new data.
- Does not support direct application of machine learning models.
- The platform is no longer maintained.

3.3 Review of Existent Platforms

By taking a closer look into the platforms described in the section 3.2, it is evident that each of the platforms has its own features and limitations. However, it's clear that none of them provides a complete solution for medical data storage, management, and analysis.

The common limitations of the existing platforms include:

- Data types limited to a specific area, such as statistical data or genomic data.

- Insufficient tools for data preprocessing, which are essential for preparing data for advanced analysis.
- Limitation on using machine learning models directly on the platform.
- Geographic and regulatory challenges, particularly for platforms based outside the European Union, impacting GDPR compliance.
- Limited user-upload capabilities, restricting the growth and enrichment of datasets.

3.4 Conclusion

The current state of the art in medical data storage, management and analysis is characterized by a different set of platforms, each addressing a different set of problems. None of the platforms described in Section 3.2 fully addresses the comprehensive needs of the Breast Cancer Research Community in Europe.

It is worth noting that while the DICOM Library provides many of the features we are looking for, however, it lacks the ability to extract and preprocess data for use in machine learning models. Additionally, the platform does not support the application of machine learning models directly.

These insights highlight the need for a new platform that can address the limitations of the existent platforms.

In the next chapters will introduce the purposed Medical Data Hub platform that aims to address these limitations and provide a comprehensive solution for medical data storage, management, and analysis.

4 | Solution Proposal

The present chapter describes in detail the solution proposal to address the problem of limited access to freely available medical data and the complex regulatory landscape governing medical data in Europe, as described in chapter 2. It covers the features, system requirements, system design, implementation, support, and security measures of the solution.

4.1 Medical Data Hub

The solution is a Medical Data Hub platform, which is a web-based application that provides a centralized platform for storing, managing, and analyzing medical data, with a particular focus on cancer data, such as breast cancer. The platform is designed to be user-friendly and accessible to medical professionals, and researchers. Additionally, the platform anonymizes the data being uploaded by the users, ensuring that patient privacy and confidentiality are mantained.

4.2 Features

The Medical Data Hub features are as follows:

- **Data Storage:** The platform will store medical data in a secure and compliant way, following the rules of the GDPR. This includes data anonymization, and access controls.
- **Dynamic Data Insertion:** Users can dynamically insert various types of data, including DICOM images, patient information, and other types of data.
- **Data Visualization:** Users will be able to visualize images and other data within the platform, enabling them to explore and analyze the data.
- **Data Export:** Users will be able to export data for use in machine learning models and other analytical tools. This will support the development of predictive models and other

research applications.

- **Machine Learning Integration:** Users will be able to apply machine learning models directly within the platform.

4.3 System Requirements

The system requirements for the Medical Data Hub platform are divided into functional and non-functional requirements. The functional requirements describe the features and functionalities that the platform should provide, while the non-functional requirements describe the quality attributes of the platform.

4.3.1 Functional Requirements

The functional requirements of the Medical Data Hub platform are as follows:

- User interface for users to interact with the platform.
- RESTful Application Programming Interface (API) for external systems to interact with the platform.
- Store medical data.
- Anonymize medical data.
- Image visualization for medical images.
- Read and write DICOM files.
- Export mechanism for users to export medical data.
- Search mechanism for users to search for medical data.
- Mechanism for users to upload and create medical data.
- Mechanism for users to download medical data.
- Authentication and authorization mechanism for users to access the platform.
- Create/upload, edit, and delete medical data.
- Add dynamic fields to medical data.
- Create and manage users.
- Create and manage permissions.
- Users can apply machine learning models to medical data.
- DICOM hierarchical structure. (e.g. Patient -> Study -> Series -> Image)

4.3.2 Non-Functional Requirements

The non-functional requirements of the Medical Data Hub platform are as follows:

- Secure.
- Scalable.
- Reliable.
- Fast.
- Flexible.
- Easy to use.
- Easy to deploy.
- Easy to maintain.
- Interoperable. (e.g., use of standards like DICOM.)
- Compliant with relevant regulations.

4.4 System Design

The present section describes the system design of the Medical Data Hub platform, includes the user interface design, data design and system architecture.
The goal of system design is to build a system that is effective, reliable and maintainable [25].

4.4.1 User Interface Design

The Medical Data Hub is a web-platform that provides a user-friendly interface for users to interact with the system.

The design of the user interface is based on the following principles:

- **Consistency:** The interface should be consistent across all pages.
- **Simplicity:** The interface should be simple and easy to use.
- **Clarity:** The interface should be clear and easy to understand.

Who is the user?

The users of the interface include Doctors and Researchers. These users will mainly interact with the platform using a Laptop or a Desktop computer. The interface should be then, designed to work in a desktop environment, and it should be also responsive to work on a tablet

if needed.

Given the different level of expertise of the users interacting with the software, the interface should be designed to be simple, intuitive and easy to use.

Use case Diagram

The diagram displayed in the Figure 4.1 shows the interactions between various users and the system.

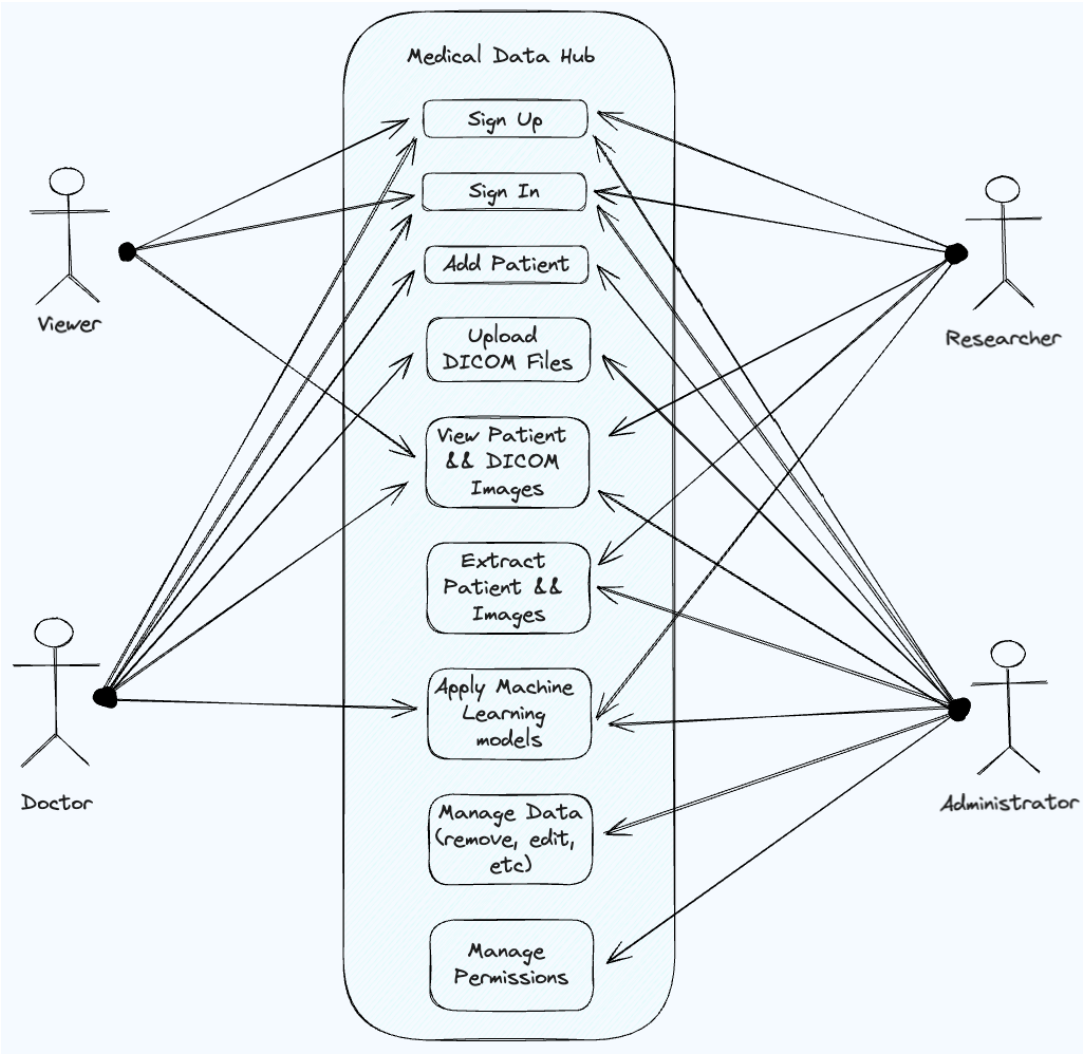

Figure 4.1: Medical Data Hub Platform use case diagram

Specifically:

- **Doctors:** Add patients, upload DICOM files, view patient and DICOM images, extract patient and image data, and apply machine learning models.
- **Administrators:** Have similar permissions as doctors but with additional administrative capabilities, that allow to manage permissions and all data in the system.

- **Viewers:** Can view patient and DICOM images.
- **Researchers:** Can view patient and DICOM images, extract patient and image data, and apply machine learning models.

All users need to sign up and sign in to the platform.

Prototype

The present section displays the prototypes of the Graphical User Interface (GUI) for the Medical Data Hub Platform. The prototypes are designed using Excalidraw [26], a tool that allows to create simple and effective prototypes.

Figures 4.2 and 4.3 display the authentication flow of the platform. The user can create an account by signing up with their email and password. Once the account is created, the user can sign in with the same credentials.

Figure 4.2: Sign Up page prototype

Figure 4.3: Sign In page prototype

Upon authentication, users are redirected to the Patient page, as shown in Figure 4.4. This page displays a table listing patients with details such as patient ID (automatically generated by the platform), gender, age, and an option to remove a patient. If the user chooses to delete a patient, a confirmation modal, shown in Figure 4.6, is displayed.

Users can also search for patients by ID and add new patients by opening the Add Patient modal, as shown in Figure 4.5. This modal allows users to input patient information such as age, gender, and additional dynamic fields using the "Add new field" option, which supports various data types including Date, Integer, and Text.

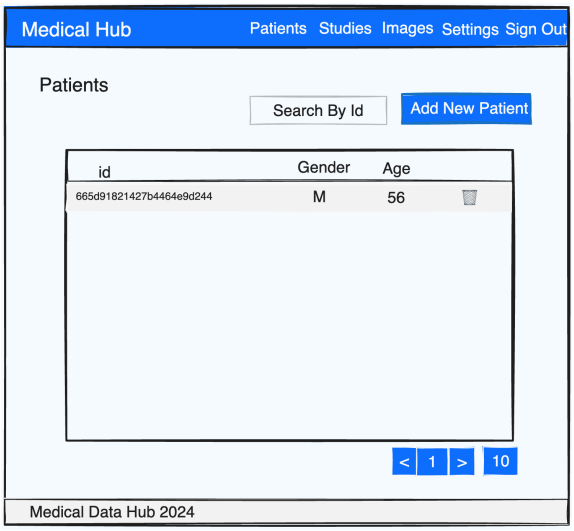

Figure 4.4: Patient page prototype

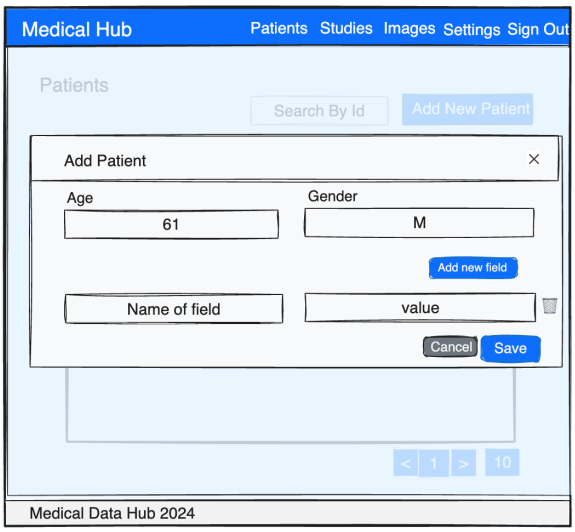

Figure 4.5: Add Patient modal prototype

Figure 4.6 displays the Delete Patient modal, which is shown when the user selects the option to delete a patient. This modal asks the user to confirm the deletion.

Figure 4.7 shows the Patient Details page, which is displayed when a patient is selected. This page allows users to view patient details, associated images, and edit patient information. Future enhancements may include adding related studies or other relevant data to this page, providing a comprehensive view of the patient's medical history.

Figure 4.6: Delete Patient modal prototype

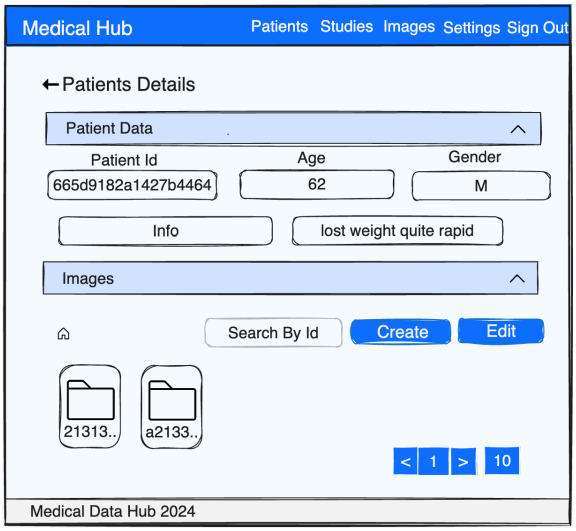

Figure 4.7: Patient Details' page prototype

The Studies' page is shown in Figure 4.8. This page lists studies associated with a patient, displaying details such as the Study ID (automatically generated by the platform), Study Date, and an option to delete a study. Users can add new studies through the Add Study modal, shown in Figure 4.9, which includes fields like Patient (dropdown list), Study Date

(date picker), and an option to add new dynamic fields.

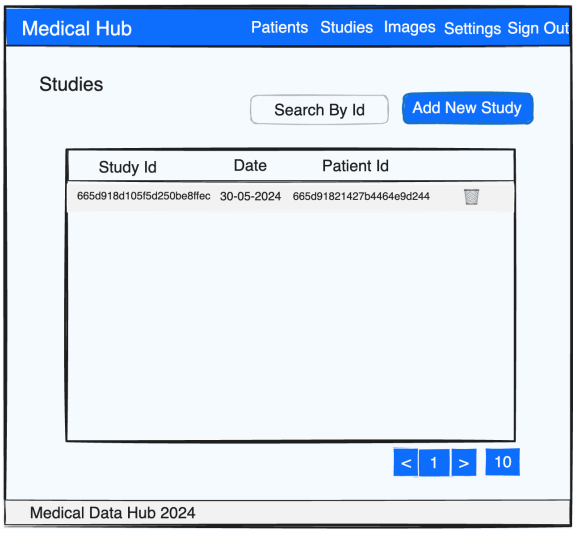

Figure 4.8: Study page prototype

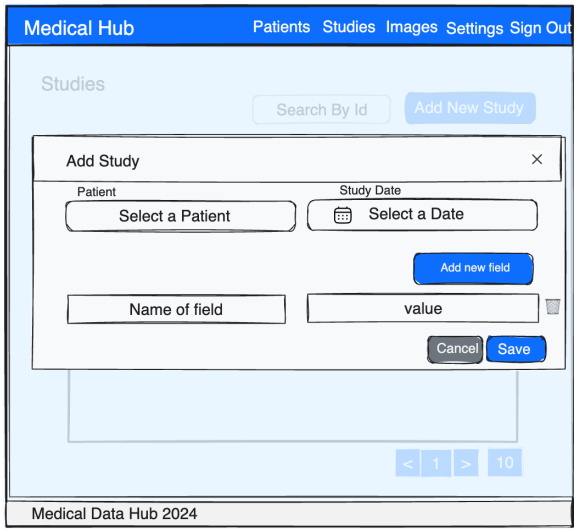

Figure 4.9: Add Study modal prototype

Figure 4.10 shows the Delete Study modal, which is displayed when a user opts to delete a study and asks for confirmation. Selecting a study opens the Study Details page, as shown in Figure 4.11, where users can view study details and associated images, and edit study information.

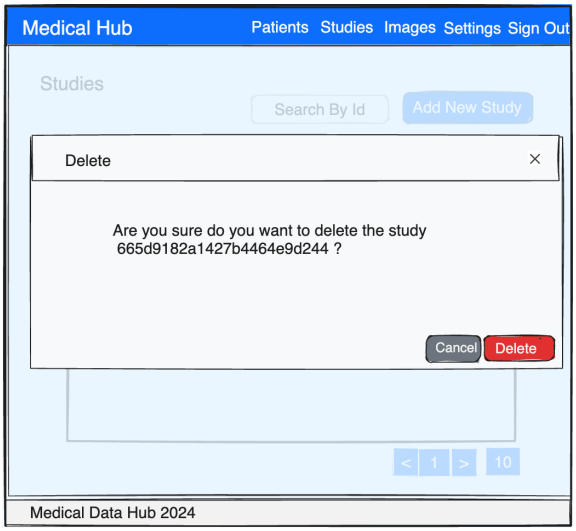

Figure 4.10: Delete Study modal prototype

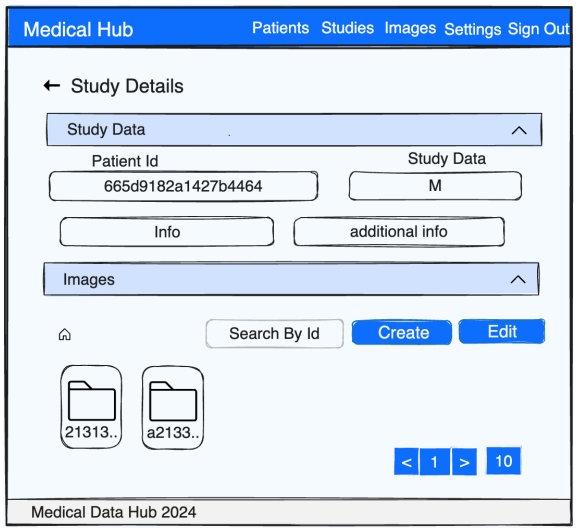

Figure 4.11: Study Details page prototype

The Images page, displayed in Figure 4.12, behaves as a file explorer, with a directory structure that follows the DICOM standard, organizing images by patient, study, and series as described in the Section 4.4.2. Users can create a new directory using the create option, which opens the Create Directory modal shown in Figure 4.13.

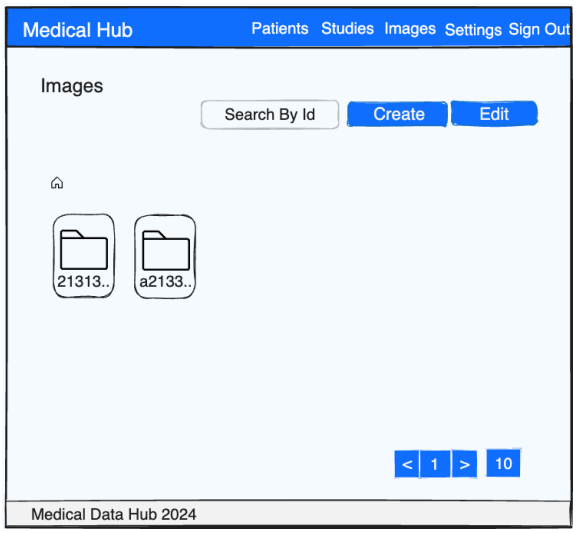

Figure 4.12: Images page with Patients prototype

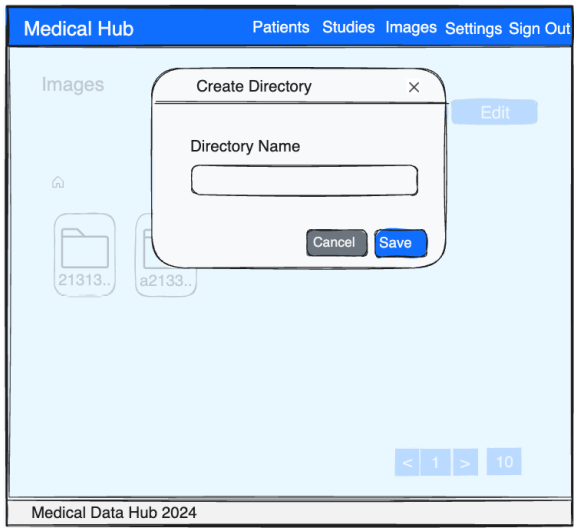

Figure 4.13: Add Directory modal prototype

Users can enable Edit mode by selecting the Edit option, allowing them to edit the name of a directory (Figure 4.15) or delete a directory (Figure 4.16).

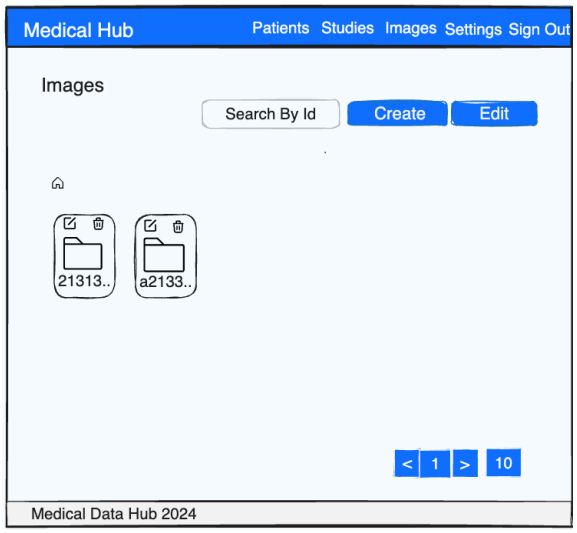

Figure 4.14: Images edit enabled prototype

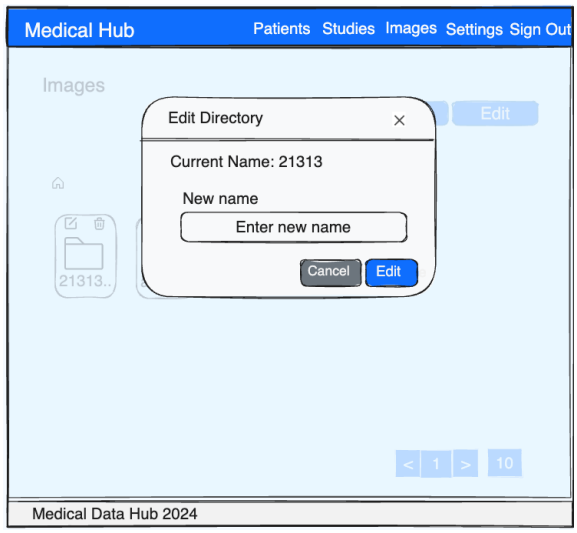

Figure 4.15: Edit Directory modal prototype

Figure 4.17 shows the Image page, where users can view images associated with a series. Users can upload images through the Upload Images option, which opens the Upload Images modal shown in Figure 4.18. The system supports the upload of multiple images at once and anonymizes the information before storing them. It supports the upload of DICOM and other image formats, storing images in TIFF format to ensure high-quality and lossless storage.

Figure 4.19 shows the Settings page, where users can manage platform settings, such as chang-

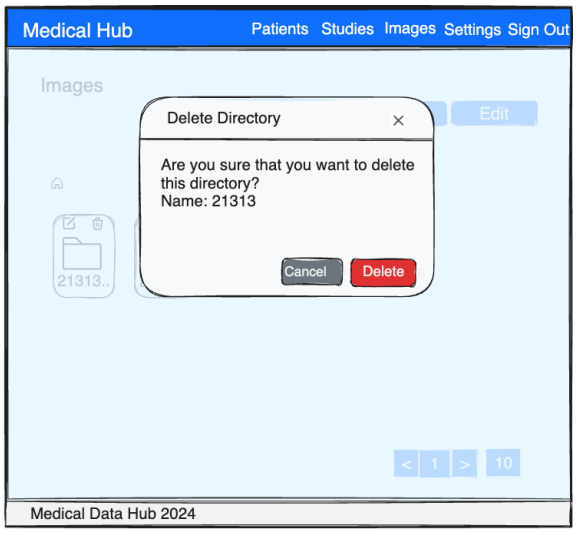

Figure 4.16: Delete Directory modal prototype

Figure 4.17: Image page with Series prototype

ing user permissions and managing users. Only users with administrative permissions have access to this page.

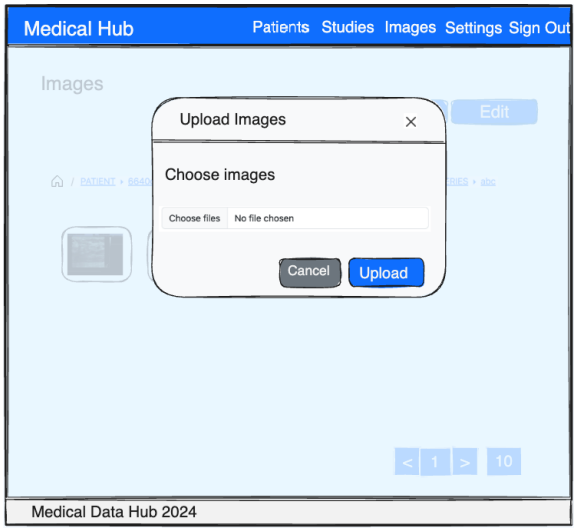

Figure 4.18: Upload Images modal prototype

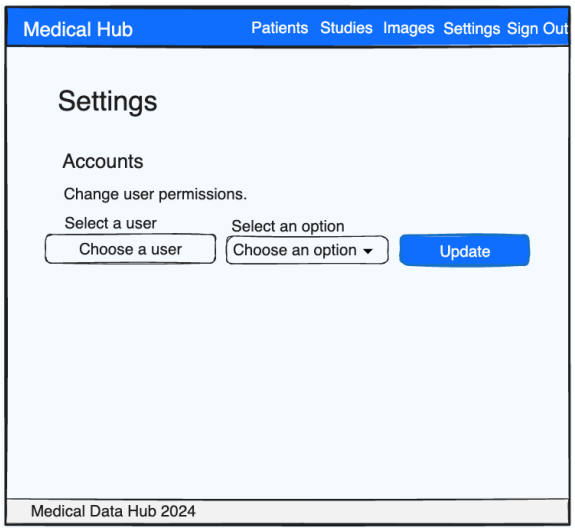

Figure 4.19: Settings page prototype

4.4.2 Data Design

The data design of the Medical Data Hub platform, focuses on minimizing the data being collected and stored, and ensuring that the data is secure and anonymized.

The data is structured in the following sequence:

- **Users:** The users of the platform.
- **Patients:** The patients.
- **Studies:** The patients studies.
- **Images:** The images collected on the studies.

Users

The Users are stored in a database, and it contains the following fields:

- **User ID:** The unique identifier for the user, which is the email.
- **Password:** The password of the user encrypted by the platform.
- **Role:** The role of the user.

Patients

The Patients are stored in a database, and it contains the following fields:

- **Patient ID:** A unique identifier for the patient. Automatically generated by the platform.
- **Age:** The age of the patient.
- **Gender:** The gender of the patient.
- **Dynamic Fields:** A list of dynamic fields that can be added by the user.

Studies

The Studies are stored in a database, and it contains the following fields:

- **Study ID:** A unique identifier for the study. Automatically generated by the platform.
- **Patient ID:** The patient that the study belongs to.
- **Study Date:** The date of the study.
- **Dynamic Fields:** A list of dynamic fields that can be added by the user.

The relationship between patients and studies is a one-to-many relationship, where a single patient can have multiple studies. It is important to note that, since MongoDB [27] is a document-based database and does not enforce relational schema, this relationship is managed at the application level.

Images

The images are stored in a file system and follow a structured hierarchy. The directory structure is designed to align with the DICOM standard, organizing images by patient, study, and series.

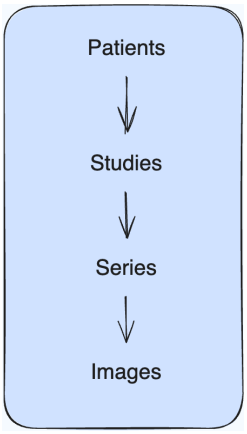

Figure 4.20: Structure of the Images File System

As illustrated in the Figure 4.20, the images are organized in directories representing patients, studies, and series. The hierarchical structure is as follows:

- **Patients:** Directory representing individual patients.
- **Studies:** Subdirectory under each patient, representing different studies conducted on that patient.
- **Series:** Subdirectory under each study, containing series of images.

The images are stored in the Series directory, with file names representing the identifiers of the patient, study, and series. The images are stored in TIFF format to ensure high-quality and lossless storage.

4.4.3 System Architecture

The system architecture of the Medical Data Hub platform is designed to be scalable, reliable, and secure. It is based on a microservices' architecture, where different components of the system are decoupled and communicate through APIs. This modular design allows for easy scaling and maintenance of the system.

The microservices' architecture was chosen for several reasons:

- **Scalability:** Individual services can be scaled independently, allowing the system to handle increased load.

- **Reliability:** The architecture ensures that the system is reliable and fault-tolerant.
- **Maintainability:** Each service can be maintained, updated, and deployed independently.
- **Flexibility:** Different technologies can be used for different services, allowing the system to leverage the most suitable tools for each specific task.
- **Agility:** The architecture enables rapid development and deployment of new features and services.
- **Security:** The architecture improves security by isolating services and enforcing strict access controls.

One of the significant advantages of this architecture is its ability to integrate new technologies seamlessly as the system evolves. This flexibility ensures that new innovations can be incorporated without disrupting existing components. This was a key consideration in the design of the Medical Data Hub platform, as it is expected to evolve over time and incorporate new technologies, such as machine learning models and data analysis tools.

The system is divided into the following services:

- **Frontend:** The user interface of the platform.
- **API Gateway:** The gateway that routes requests to the appropriate microservices and handles authentication, providing a unified access point for all clients.
- **Patients Service:** Manages patient data, including creating, reading, updating, and deleting patient records.
- **Studies Service:** Manages data related to medical studies, including study creation, reading, updating, and deletion.
- **Images Service:** Handles the storage, retrieval, and processing of medical images.
- **Export Data Service:** Provides functionality for exporting data in various formats for further analysis or external use.
- **Machine Learning Service:** Integrates and applies machine learning models to analyze medical data and provide predictive analytics.

The use of different technologies for various services ensures that the platform can adapt to the latest advancements and integrate new tools and techniques as they become available. This approach not only enhances the platform's performance but also ensures its long-term viability and adaptability.

The Figure 4.21 illustrates the system architecture of the Medical Data Hub platform, showing

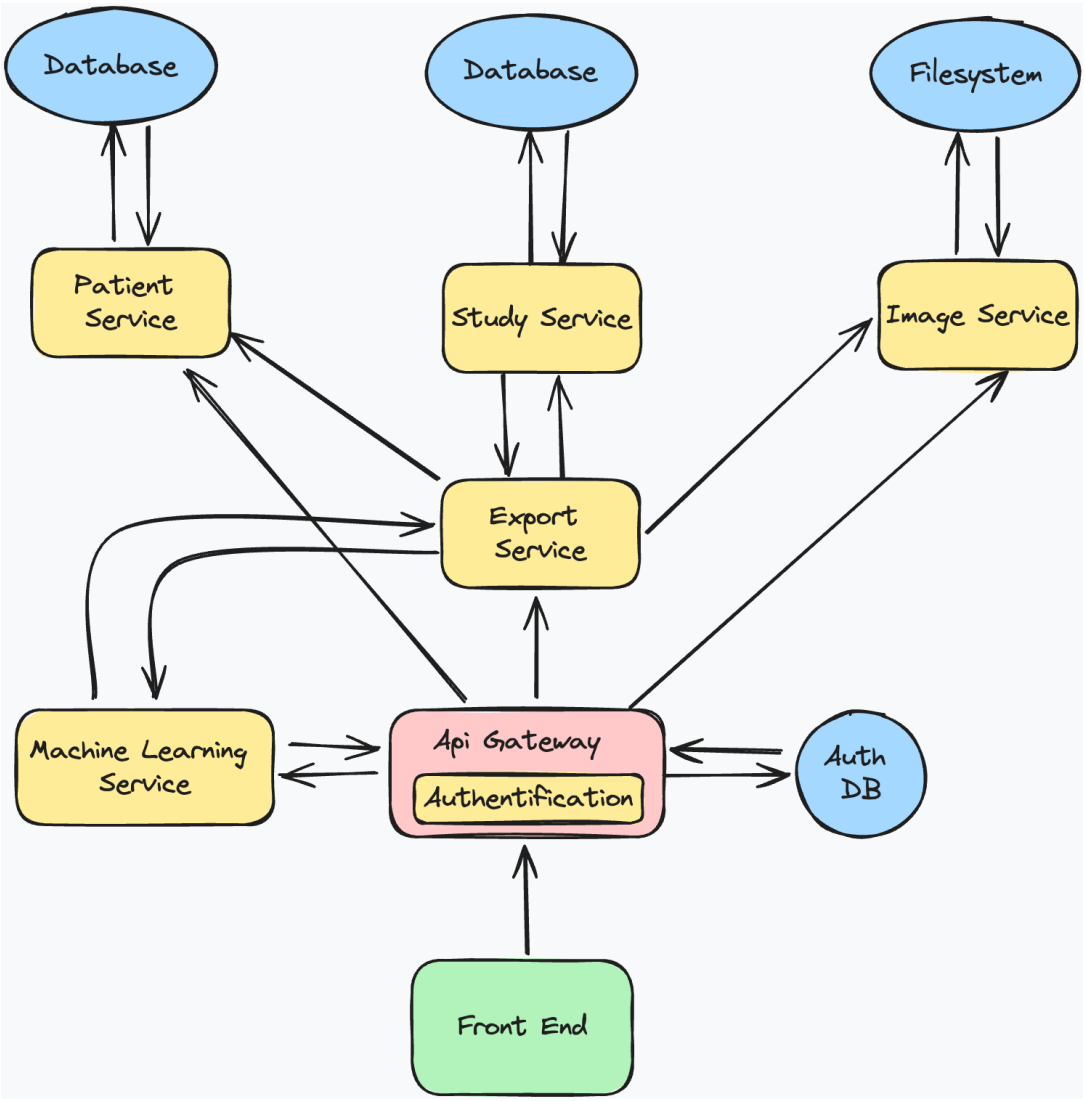

Figure 4.21: System Architecture of the Medical Data Hub Platform

the different services and their interactions. The Machine Learning Service requires a database to store the models and the results of the predictions, this database is not shown in the figure, but it is part of the system architecture.

Trade-offs

While the microservices' architecture offers several advantages, it also comes with trade-offs. Some trade-offs to consider when using a microservices' architecture include:

- **Complexity:** Managing multiple services can be complex and require additional effort to coordinate and maintain.
- **Latency:** Communication between services can introduce latency, especially in distributed systems.
- **Data Consistency:** Ensuring data consistency across services can be challenging, as each

service may have its own database or data store.

- **Security:** Securing communication between services and enforcing access controls can be more complex in a microservices' architecture.
- **Testing:** Testing individual services and ensuring they work together can be more challenging than testing a monolithic application.

Despite these trade-offs, this architecture provided the best balance for the platform, primarily due to the agility that provides to the system, as it allows to easily integrate new technologies and scale the system as needed, especially if we compare it with a monolithic architecture, where the system would be more rigid and less adaptable to change. This allows to use different Databases, different programming languages and different technologies for each service, which is a significant advantage for the Medical Data Hub platform, as it is expected to evolve and incorporate new technologies over time.

4.5 System Implementation

The following section describes the implementation details of the Medical Data Hub platform, including the development environment, and the overview of the technologies used. For a detailed description of the technologies refer to the chapter 5. Jira [28] is utilized as a project management tool to track the progress of the project and manage the tasks effectively.

4.5.1 Development Environment

The development environment includes the following components:

- **Version Control System:** Git [29] is used for version control, with repositories hosted on GitHub [30].
- **Integrated Development Environment (IDE):** Visual Studio Code [31] is used as the primary IDE.
- **Containerization:** Docker [32] is used for containerizing the application, ensuring consistency across different environments.

4.5.2 Technologies Used

The Medical Data Hub platform is developed using the following technologies:

- **Frontend:** Built with Next.js [33], React [34], Typescript [35], React-bootstrap [36].

- **Backend:** Implemented using Node.js [37] with Express.js [38] and Typescript for the API Gateway, and Python [39] with Fast API [40] for microservices.
- **Database:** MongoDB is used as the primary database for storing patient and study data.
- **Authentication:** JSON Web Tokens (JWT) are used for authentication and authorization. Lightweight Directory Access Protocol (LDAP) is used for storing user credentials and roles.
- **Storage:** The file system is used for storing DICOM images.
- **Containerization:** Docker is used for containerizing the application, ensuring consistency across different environments.

For additional details about the technologies used refer to the chapter 5.

4.5.3 Service Implementation

The system is implemented as a set of services, with each one responsible for a specific function. This section presents the frontend service, API Gateway service, and the microservices responsible for managing patient data, study data, and DICOM images. The implementation details for each service are as follows:

Frontend Service

The Frontend Service is implemented using Next.js, React, and Typescript. It interacts with the API Gateway service through Restful APIs.

The main features of the frontend service include:

- User authentication and authorization.
- Form interfaces for adding and managing patients and studies.
- Tools for uploading and viewing DICOM images.
- Searching and filtering patient and study data.

Other features like exporting data and machine learning integration are not implemented in the current version of the system.

API Gateway Service

The API Gateway Service is implemented using Node.js with Express.js and Typescript. It acts as a proxy for the Frontend Service, routing requests to the appropriate microservices. It is also responsible for the authentication of the user.

The API Gateway is responsible for the following tasks:

- User authentication and partial authorization.
- Routing requests to the appropriate microservices.
- Error handling and logging.

The role of the API Gateway Service in the authorization process is to check if the user has a valid session token before allowing access to the microservices.

The API Gateway interacts with the following services:

- **Patient Service:** Responsible for managing patient data.
- **Study Service:** Responsible for managing study data.
- **Image Service:** Responsible for managing DICOM images.
- **Export Service:** Responsible for exporting data.
- **Machine Learning Service:** Responsible for machine learning tasks.

Patient Service

The Patient Service is implemented using Python with Fast API and uses MongoDB as the database.

The main features of the Patient Service include:

- Create, Read, Update, Delete (CRUD) operations for managing patient data.
- Searching and filtering patient data.
- Validation of patient data.
- Authorization checks for user roles.

Study Service

The Study Service is implemented using Python with Fast API and uses MongoDB as the database.

The main features of the Study Service include:

- CRUD operations for managing study data.
- Searching and filtering study data.
- Validation of study data.
- Authorization checks for user roles.

Image Service

The Image Service is implemented using Python with Fast API and uses the file system for storing DICOM images.

The main features of the Image Service include:

- Uploading and storing DICOM images.
- Retrieving and displaying DICOM images.
- Searching and filtering images.
- Anonymizing DICOM images.
- Authorization checks for user roles.

The Export and Machine Learning Services are not implemented in the current version of the system.

4.5.4 Deployment and Testing

The system is deployed using Docker containers. This ensures consistency across different environments and simplifies the deployment process.

The current system doesn't have automated tests, but manual testing was performed to ensure the system's functionality.

4.6 System Support and Security

The support and security of the Medical Data Hub platform are critical aspects of the system, ensuring that the system is reliable and secure.

This section discusses the measures taken to provide robust support and security for the platform.

4.6.1 System Support

Technical Support

The system support will be provided by the development team, who are responsible for maintaining the system and assisting users. As the platform is still in the early stages, support is limited but will expand as the system matures.

Maintenance and Updates

To ensure the system remains functional and up-to-date, the development team will periodically release updates and perform maintenance tasks.

- **Regular Updates:** The team will schedule regular updates to fix bugs, implement new features, and improve system performance.
- **Backup and Recovery:** Periodic backups will be conducted to prevent data loss and ensure data recovery in case of system failure.

4.6.2 System Security

The system security is one of the most important aspects of the system, as it will store sensitive data and will be accessed by multiple users.

Data Protection

- **Encryption:** Use Hypertext Transfer Protocol Secure (HTTPS) to ensure that the data is secure during transmission.
- **Access Controls:** Implement role-based access controls (RBAC) to ensure that users have access solely to the data and functionalities necessary for their roles.
- **Data Anonymization:** Apply data anonymization techniques to protect patient privacy while enabling data analysis and research.

The system has a login system that requires a username and password to access the system. The password is stored in the database as encrypted hash, to ensure that the if the database is compromised, the passwords are not exposed.

Authentication and Authorization

- **JWT Tokens:** Use JSON Web Tokens (JWT) for secure authentication and authorization of users.
- **LDAP Integration:** Utilize LDAP for managing user credentials and roles, ensuring secure and scalable user management.

Authentication is performed at the API Gateway level, while authorization is handled at the microservices level. The API Gateway validates the JWT token, performing partial authorization, and forwards the request to the appropriate microservice. This ensures that microservices are secure and accessible only to authorized users.

4.6.3 Monitoring and Logging

Monitoring and logging are essential to ensure that the system is running smoothly and to identify any potential issues that may occur.

The system has a logging system that logs all the requests and responses to the system. However, the system lacks a monitoring system to monitor the system health and performance. Currently, this isn't an issue, but as the system grows, a monitoring system would be required to ensure that the system is running smoothly.

4.6.4 Conclusion

As the system matures and grows, the support will need to grow and adapt to the needs of the users. Currently, the system is still in the early stages and the support is very minimal; as the platform grows and more users start using the system, the support will need to grow to ensure that the system is reliable and secure.

Security is a crucial aspect of the system, ensuring data protection and user privacy. Various security measures such as JWT, RBAC have been implemented to secure the system and protect data. Continuous improvements and adaptations will be necessary as the system grows and evolves, maintaining the integrity and security of the platform.

5 | Technologies Used

The present chapter provides a comprehensive overview of the technologies used in the Medical Data Hub platform. It details the rationale behind the technologies selection, highlighting the criteria used in the decision-making process, compares the chosen technologies with other available options, and concludes with a summary.

5.1 Overview of Used Technologies

This section provides a summary of the technologies used in the Medical Data Hub, including web frameworks, data storage solutions, security protocols, data anonymization techniques, and others.

5.1.1 Frontend Technologies

Table 5.1 provides an overview of the main frontend technologies used in the Medical Data Hub.

Technology	**Description**
Next.js	A React Framework for building full-stack web applications.
Typescript	A typed programming language that builds on JavaScript.
HTML/CSS	The standard markup language for creating web pages [41].
React-Bootstrap	A popular React UI framework.
Viewer.js	A JavaScript library for image viewer.
React-grid-layout	A JavaScript library for creating beautiful grids [42].

Table 5.1: Frontend Technologies

Additionally, there were some small libraries being used for specific functionalities, such as:

- **react-datepicker**: A library for having a beautiful date picker [43].
- **react-bootstrap-typeahead**: A library for having a typeahead search bar [44].

- **eslint**: A library for linting the code, helps to find and fixing problems in the code [45].

5.1.2 Backend Technologies

Table 5.2 provides an overview of the main backend technologies used in the Medical Data Hub.

Technology	Description
Node.js	A JavaScript [46] runtime built on Chrome's V8 engine.
Python	A high-level programming language.
Typescript	A typed programming language that builds on JavaScript.
Express.js	A lightweight and adaptable framework for building web applications with Node.js.
FastApi	A modern, fast (high-performance), web framework for building APIs with Python based on standard Python type hints.
Pymongo	A Python library that allows to interact with MongoDB an object modeling tool [47].
Ldapjs	A pure JavaScript LDAP client library for Node.js [48].
http-proxy-middleware	A library for proxying requests [49].
Pydicom	A Python library for managing Digital Imaging in Medicine [50].
Presidio Image Reductor	A Python library for anonymizing medical images [51].

Table 5.2: Backend Technologies.

Additionally, there were other libraries being used for specific functionalities, such as:

- **dotenv**: A library for loading environment variables from a .env file [52].
- **cors**: A library for enabling Cross-Origin Resource Sharing (CORS) with various options [53].
- **body-parser**: A library for parsing incoming request bodies [54].
- **cookie**: A library for basic HTTP cookie parser and serializer for HTTP servers [55].
- **jsonwebtoken**: A library for creating and verifying JSON Web Tokens [56].
- **morgan**: A library for HTTP request logger middleware for Node.js [57].
- **swagger-ui-express**: A library for delivering automatically generated Swagger UI API documentation through an Express server [58].
- **swagger-jsdoc**: A library for generating Swagger/OpenAPI 3.0 specifications from JSDoc comments [59].

- **pydantic**: A library for validating data and managing settings with Python type hints [60].
- **uvicorn**: An ASGI web server implementation for Python [61].
- **numpy**: A library for numerical computing with Python [62].
- **pillow**: A library for image processing with Python [63].
- **python-multipart**: A streaming multipart parser for Python, licensed under Apache2 [64].

5.1.3 Data Storage Technologies

Table 5.3 provides an overview of the main data storage technologies used in the Medical Data Hub.

Technology	Description
MongoDB	A source-available, cross-platform, document-oriented database program.
LDAP	A protocol for accessing and maintaining distributed directory information services over an IP network [65].
File System	A structure used by an operating system to organize and manage files on a storage device [66].

Table 5.3: Data Storage Technologies

5.1.4 Security Technologies

Table 5.4 provides an overview of the main security technologies used in the Medical Data Hub.

Technology	Description
JWT	A standard for creating access tokens [67].
Pyjwt	A Python library which allows you to encode and decode JSON Web Tokens [68].
Bcrypt	A password hashing function [69].

Table 5.4: Security Technologies

5.1.5 Additional libraries and tools

Table 5.5 describes the libraries and tools that significantly impact the overall project, covering both frontend and backend aspects, including deployment, tracking, and other essential functions.

Technology	Description
Docker	An open platform for developing, shipping, and running applications inside containers [32].
Git	A distributed version-control system for tracking changes in source code during software development.
Github	A provider of Internet hosting for software development and version control using Git.
Gitkraken	A Git GUI client [70].
Jira	A proprietary issue tracking product developed by Atlassian.
Visual Studio Code	A source-code editor developed by Microsoft.
Orbstack	A platform for managing and deploying applications [71].
Postman	API platform for developers to design, build, test, and collaborate on APIs [72].
Confluence	A collaboration wiki tool used to help teams collaborate and share knowledge efficiently [73].
Pip	A package installer for Python [74].
NPM	A package manager for JavaScript [75].
Excalidraw	An open source virtual hand-drawn style whiteboard.
Phpldapadmin	A web-based LDAP client [76].
Mongo-express	A web-based MongoDB admin interface [77].
http-status-codes	A library that helps to read HTTP status codes [78].

Table 5.5: Other Technologies

5.2 Choosing Critea

The technologies used in the Medical Data Hub were chosen based on the following criteria:

- **Purpose**: The technologies should meet the requirements and goals of the platform.
- **Easy of Use**: The technologies should be easy to use to ensure that the platform can be developed quickly.
- **Community Support**: The technologies should have a large community support to ensure that the platform can be maintained and updated in the future.
- **Documentation**: The technologies should have good documentation to ensure that the platform can be developed and maintained easily.
- **Update and Maintainability**: The technologies should be updated and maintained regularly to ensure that the platform can be secure and performant.
- **Scalability**: The technologies should be scalable to handle a large number of users and data.
- **Security**: The technologies should be secure to ensure that the data is protected.
- **Performance**: The technologies should be fast and stable to ensure that the platform can handle a large number of users and data.

The current section outlines the decision taken for each major technology used in the Medical Data Hub.

5.2.1 Frontend Technologies

Next.js

Next.js is a React framework that allows building full-stack web applications. It was chosen due to his flexibility, features and easy of use. It provides a lot of features out of the box, such as server-side rendering, static site generation, and API routes. Next.js also has a large growing community and good documentation, which makes it easy to learn and use. Additionally, Next.js is updated regularly and is maintained by Vercel, which ensures that the platform can be used long term. The technology is also scalable, fast and secure, which makes it a good choice for the Medical Data Hub.

Typescript

Typescript is a typed programming language that builds on JavaScript. It was chosen to be used in the frontend due to the typing feature that it provides for JavaScript. This allows for catching errors early in the development process, making the code more robust, easier to maintain and understand. Additionally, Typescript has a growing large community, good documentation, and is updated regularly, which makes it easy to learn, use, and it's a good choice for the Medical Data Hub.

React-Bootstrap

React Bootstrap is a popular React UI framework, it was chosen due to his easy of use, quickly and beautiful layouts. This technology provides lot of responsive components that uses Bootstrap, which speeds up the development process. Additionally, it has a large community and good documentation, which makes it easy to learn and use.

Viewer.js

Viewer.js is a library that allows to visualize images and manipulate them in a simple and easy way. Its features such as zoom, rotate, flip, and crop make it a good choice for the Medical Data Hub. It has a great popularity and good documentation, which makes it easy to learn and use. These features are appropriate to visualize the medical images in the Medical Data Hub and to provide a good user experience.

React-grid-layout

React-grid-layout is a library that allows to create beautiful grids in a simple and easy way. This library was used to create a beautiful grid layout for the images, and to simulate a File System. This is one of the few libraries that provided the features needed for the Medical Data Hub.

5.2.2 Backend Technologies

Node.js and Express.js

Node.js and Express.js were chosen as the backend technologies for the API Gateway. The reason for this choice is that Node.js is a fast, scalable, and secure runtime for JavaScript, and Express.js is a lightweight and adaptable web application framework designed for Node.js. This is a good choice for creating an API Gateway that works as a proxy between the frontend and the microservices on the backend. It has the capabilities to integrate with different

frameworks and libraries, such as ldap.js, http-proxy-middleware, JWT, and bcrypt, making a perfect choice to handle the authentication of the users and act as a proxy for the microservices. Additionally, it has a large community, good documentation, and is updated regularly, which makes it easy to learn, use, and a good choice for the Medical Data Hub.

Python and FastApi

Python and FastApi were chosen as the backend technologies for the microservices. The reason for this choice is that Python is a high-level programming language that is easy to learn and use. FastApi is a modern, fast, web framework for building APIs with Python based on standard Python type hints. It provides a lot of features out of the box, such as data validation, settings management, auto-generated API docs, and most important it provides access to the Python ecosystem, which is rich in libraries and tools, such as Pydicom, Presidio Image Redactor, and Pymongo, allowing easy integration with MongoDB, the DICOM standard and health data anonymization. Python is one of the most popular ecosystem for data science and machine learning, which makes it a good choice for the Medical Data Hub. Additionally, Python and FastApi have a large community, good documentation, and are updated regularly.

5.2.3 Data Storage Technologies

MongoDB

For storing the medical metadata, MongoDB was chosen due to his flexibility, scalability, and performance.

MongoDB is a NoSQL database based on a document-oriented data model, which provides significant advantages over relational databases.

This flexibility is crucial for the Medical Data Hub, as it allows storing different types of data, such as patient information, study information and other relevant data. This schema-less design allows adapting quickly and easy to changes in the data model, which is crucial as new requirements and features emerge.

LDAP

For storing the user information, LDAP was chosen due to his flexibility, features and security. LDAP is a protocol used for accessing and managing distributed directory information services over an IP network, which provides a centralized way to store and manage user information. This is crucial for the Medical Data Hub, as it allows storing user information in a

secure and scalable way, and to authenticate users against the LDAP server. By centralizing user information, LDAP facilitates efficient management of user credentials and access controls, thereby enhancing the platform's overall security and integrity.

File System

For storing the medical images, File System was chosen due to his simplicity, performance and scalability.
File System is a structure used by an operating system to organize and manage files on a storage device, which provides a simple and efficient way to store and retrieve medical images. This is crucial for the Medical Data Hub, as it allows store and manage large volumes of medical images in a scalable and efficient way. By leveraging the File System, the Medical Data Hub can handle a huge amount of medical images, ensuring quick access and retrieval while maintaining high performance and reliability. This data storage solution is ideal for the Medical Data Hub, as it provides a cost-effective and scalable way to store and manage medical images.

5.2.4 Security Technologies

JWT

JWT is a standard for creating access tokens that can be used to authenticate users and authorize access to resources. It provides a secure and efficient way to manage user authentication and access control, which is crucial for the Medical Data Hub.
By using JWT, the Medical Data Hub can ensure that user credentials are securely stored and transmitted, and that access to resources is properly controlled and monitored.

Bcrypt

Bcrypt is a password hashing function that provides a secure and efficient way to store user passwords. It ensures that user passwords are securely stored and protected from unauthorized access, which is crucial for the Medical Data Hub. By using Bcrypt, the Medical Data Hub can ensure that user passwords are properly encrypted and protected, and that user authentication is secure and reliable.

5.2.5 Additional Libraries and tools

Docker

Docker is an open platform for developing, shipping, and running applications inside containers [32]. It provides an easy way to deploy applications independently of the underlying infrastructure, being crucial to Medical Data Hub platform, as the application can be deployed in different environments for development, testing, and production without having any issue of compatibility. As well, it provides an easy way to deploy the microservices from the Backend and the Frontend in a single environment, which makes it a good choice for the Medical Data Hub.

Git

Git is a distributed version-control system for tracking changes in source code during software development. It provides an efficient way to manage and track changes in the codebase. By using Git, the Medical Data Hub can ensure that the codebase is properly versioned and maintained, and that changes are properly tracked and documented.

Microsoft Presidio

Microsoft Presidio is a library that provides fast identification and anonymization modules for text and images. This library has the features required to anonymize medical images and text, which is crucial for the Medical Data Hub.

Pydicom

Pydicom is a library that provides tools for managing Digital Imaging in Medicine. The library provides the features required to read, write, anonymize, and manipulate DICOM files, which is crucial for the Medical Data Hub.

5.3 Discussion and Comparison of Technologies

The present section makes a comparison between the most popular frontend frameworks and libraries, highlighting the strengths, weaknesses, and use cases of each technology. It also discusses the decision-making process and the reasons for choosing the technologies used in the Medical Data Hub.

5.3.1 Frontend Technologies

In addiction to Next.js, which was chosen for the implementation of the system, the following technologies were studied: React, Angular [79] and Vue.js [80]. Table 5.6 presents the strengths, weaknesses and use cases for each.

Technology	Strengths	Weaknesses	Use Cases
Next	-Uses React -Improved performance -Server-side rendering (SSR) -Static site generation (SSG) -SEO -API routes	-No built-in state management -Complexity	-Websites needing SEO and fast server-side rendering
React	-Scalability -Virtual Document Object Model (DOM) -Large ecosystem -Flexibility -Fast development	-Not a full framework -Disperse documentation	-Websites needing flexibility and customization -Projects where SSR is not a priority
Angular	-Comprehensive framework -Built-in tools for routing, forms, state management	-Verbose and complex -Hard to learn	-Large-scale enterprise applications - Projects with complex requirements
Vue.js	-Flexibility -Easy to learn -Reactive Data Binding -Virtual DOM	-Lack of scalability -Small ecosystem	-Small to medium-sized projects -Projects with tight deadlines

Table 5.6: Frontend Technologies Comparison

All the frontend technologies compared have strong communities, with React, Angular, and Vue.js being the most popular in this order.

The choice of Next.js over the other technologies was made due to its combination of React's features with the added benefits of server-side rendering and static site generation that Next.js provides.

React is a widely used library with significant advantages, but it lacks some features crucial for the Medical Data Hub, such as server-side rendering, static site generation, and built-in API routes. These features are essential for the Medical Data Hub as they offer better performance,

Search Engine Optimization (SEO), and flexibility, which are critical for the platform.

While Angular and Vue.js are also excellent frameworks, they were not the optimal choice for the Medical Data Hub. For instance, Vue.js, despite being a great framework, lacks scalability and has a smaller ecosystem. Angular, on the other hand, is quite verbose and complex, making it harder to learn and use effectively.

5.3.2 Backend Technologies

In addiction to Python, which was chosen for the implementation of the system, the following technologies were studied: FastAPI, Flask [81], Django [82], and Express.js. Table 5.7 presents the strengths, weaknesses, and use cases of each.

The API Gateway could be built using different backend languages, such as Python, Ruby, Java, etc. However, the choice was made based on the simplicity and flexibility of Node.js and Express.js, which are widely used in the industry. The JavaScript language was chosen due to the frontend being developed in Next.js, which is also based on JavaScript, allowing for better integration between the frontend and the backend. Then the simplicity and configuration of the Express.js allows for a fast development of the API Gateway, which is crucial for the Medical Data Hub.

Technology	**Strengths**	**Weaknesses**	**Use Cases**
FastAPI	-High performance -Easy to use -Automatic interactive API documentation -Based on standard Python type hints	-Less mature ecosystem compared to Flask and Django	-Suitable for building fast and efficient APIs -Projects needing automatic documentation
Flask	-Lightweight and flexible -Large ecosystem -Easy to learn	-Limited built-in features -Requires extensions for full functionality	-Simple web applications -Prototypes and Minimum viable products (MVP)
Django	-Full-featured framework -Built-in admin interface -Strong community support	-Monolithic structure -Can be overkill for simple applications	-Large-scale web applications -Projects needing a robust admin interface
Express.js	-Minimalistic and flexible -Large ecosystem -Fast development	-Requires additional middleware for full functionality	-RESTful APIs -Single-page applications (SPA)

Table 5.7: Backend Technologies Comparison

The choice of FastAPI over other technologies was due to its high performance, ease of use, and automatic interactive API documentation. FastAPI is a modern, fast web framework for building APIs with Python based on standard Python type hints. It offers numerous features out of the box, such as data validation, settings management, and auto-generated API docs. Furthermore, it provides easy integration with MongoDB, the DICOM standard with the use of Pydicom. Python's popularity in data science and machine learning also makes it an excellent choice for the Medical Data Hub.

Flask is also a good framework for building microservices. However, FastAPI was chosen for the Medical Data Hub due to its superior performance and automatic interactive API documentation.

In the future, the microservices' architecture allows for the use of different frameworks, languages, and tools, such as Ruby [83] or Java [84], depending on the specific needs of the software. This decision ensures that the platform can grow and integrate new technologies to suit different use cases.

5.3.3 Data Storage Technologies

MongoDB and MySQL [85] are two prominent database technologies, each with its own strengths and weaknesses. Table 5.8 summarizes the two technologies based on their strengths, weaknesses, and use cases.

Atomicity, consistency, isolation, and durability (ACID) compliance are essential for complex transactions, making MySQL a Structured Query Language (SQL) a good choice for applications with such requirements. However, MongoDB's schema-less design, horizontal scalability, and flexible data model make it ideal for projects handling diverse and unstructured data, such as the Medical Data Hub platform. It allows storing different types of data, such as patient information, study information, and other relevant data.

Technology	Strengths	Weaknesses	Use Cases
MongoDB	-Schema-less design -Horizontal scalability -Flexible data model -Efficient for unstructured data	-Not ideal for complex transactions -Less mature querying capabilities compared to SQL databases	-Applications requiring flexibility and scalability -Projects handling diverse and unstructured data
MySQL	-Strong ACID compliance -Mature ecosystem -Powerful querying capabilities -Suited for complex transactions	-Rigid schema design -Vertical scalability limitations -Less efficient for handling unstructured data	-Applications with complex transactional requirements -Projects needing structured data management

Table 5.8: Data Storage Technologies Comparison

5.3.4 Security Technologies

Security is extremely important for the Medical Data Hub platform. LDAP and OAuth [86] are two widely used authentication technologies, each with distinct advantages and disadvantages. Table 5.9 summarizes their strengths, weaknesses and use cases.

Technology	Strengths	Weaknesses	Use Cases
LDAP	-Centralized directory services -Strong authentication and authorization -Widely used in enterprise environments	-Complex setup -Primarily used for internal network authentication	-Applications requiring centralized user management -Projects needing strong access control
OAuth	-Standardized protocol for authorization -Supports third-party application access -Widely adopted for web and mobile applications	-Primarily focuses on authorization rather than authentication -Requires integration with other services for full user management	-Applications needing secure delegated access to resources -Projects requiring third-party integration

Table 5.9: Security Technologies Comparison

LDAP was chosen over OAuth as listen in Table for its strong authentication and authorization capabilities, which are essential for securely managing medical data and user access within the Medical Data Hub platform.

5.4 Conclusion

In conclusion, the technologies used in the Medical Data Hub were chosen by the need to build a robust, scalable, and secure platform that meets the requirements and goals of the proposal.

Every technology was chosen for its specific strengths and how well integrates with the overall architecture of the platform, ensuring that the platform can be developed quickly, maintained easily, and used long term.

6 Results and Discussion

This chapter presents the Medical Data Hub platform as a result, displaying the final product and discussing the platform's features and limitations.

6.1 Medical Data Hub Platform

The Medical Data Hub Platform is a web-based application that provides a user-friendly interface for researchers to access and analyze medical data. The platform is designed to be easy to use and intuitive, with a focus on providing a seamless experience for users. The platform is built using modern web technologies, including Next.js, React, Node.js, MongoDB, microservices architecture, and others technologies, to provide a fast and responsive experience for users. The frontend interacts with a RESTful API, which communicates with an API Gateway that handles the communication between the frontend and backend microservices.

The current version of the platform provides the following features:

- **Data Storage:** The platform stores medical data in a secure and compliant way, following the rules of the GDPR. This includes DICOM files and images anonymization, and access controls (authentication and authorization).
- **Dynamic Data Insertion:** Users can dynamically insert/upload various types of data, including DICOM images, patient information, and other types of data.
- **Data Visualization:** Users can visualize, filter and search images and other data within the platform.

The current version lacks some features that were planned but not implemented due to time constraints. These features include:

- **Data Export:** The ability to export data in various formats, such as Comma-separated values (CSV) [87] or JSON [88] to train machine learning models;

- **Machine Learning Integration:** The ability to perform data analysis on the platform, such as running machine learning models.

The following figures, designated with 6.*, illustrate the layout and user interface of the Medical Data Hub platform. Notably, Figure 6.18 shows a DICOM image containing patient information, while Figure 6.19 displays an Ultrasound image with data anonymized by the Medical Data Hub platform. This anonymization process removed personal information such as the patient's name, date of birth, and other sensitive details. The displayed image is a DICOM file that was uploaded to the platform and anonymized by the backend microservices.

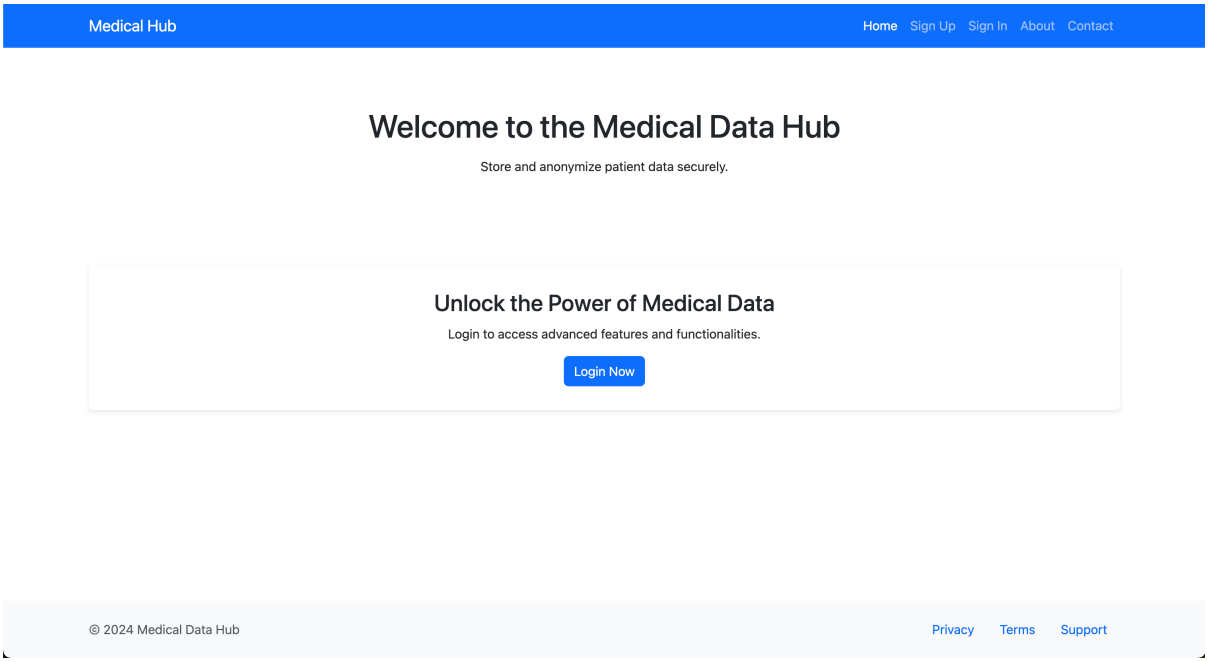

Figure 6.1: Medical Data Hub Home Page

6.2 Discussion

The platform received positive feedback from two users, both researchers, to whom the platform was displayed. The platform was praised for its user-friendly interface, ease of use, and intuitive design. The platform's features were also well-received, with users expressing interest in using the platform for their future research projects.

It is important to note that the platform is still in the early stages of development and has room for improvement.

The platform's current version lacks some important features that were planned but not implemented due to time constraints, as described previously in the Section 6.1. These features

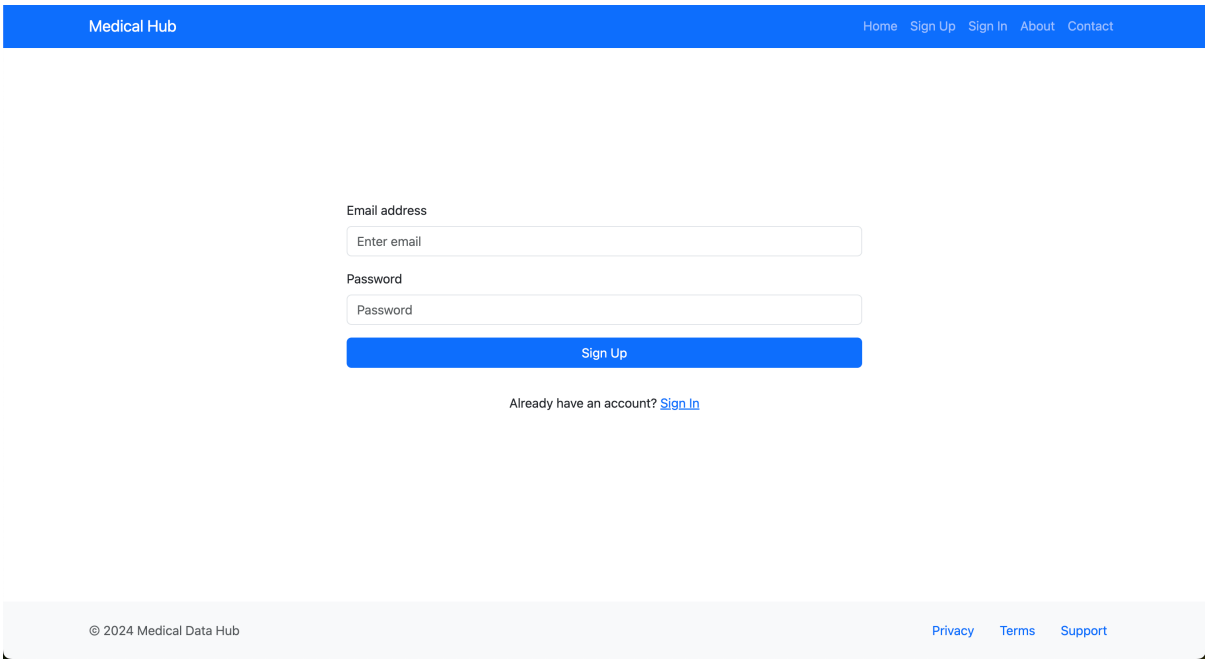

Figure 6.2: Medical Data Hub Sign Up Page

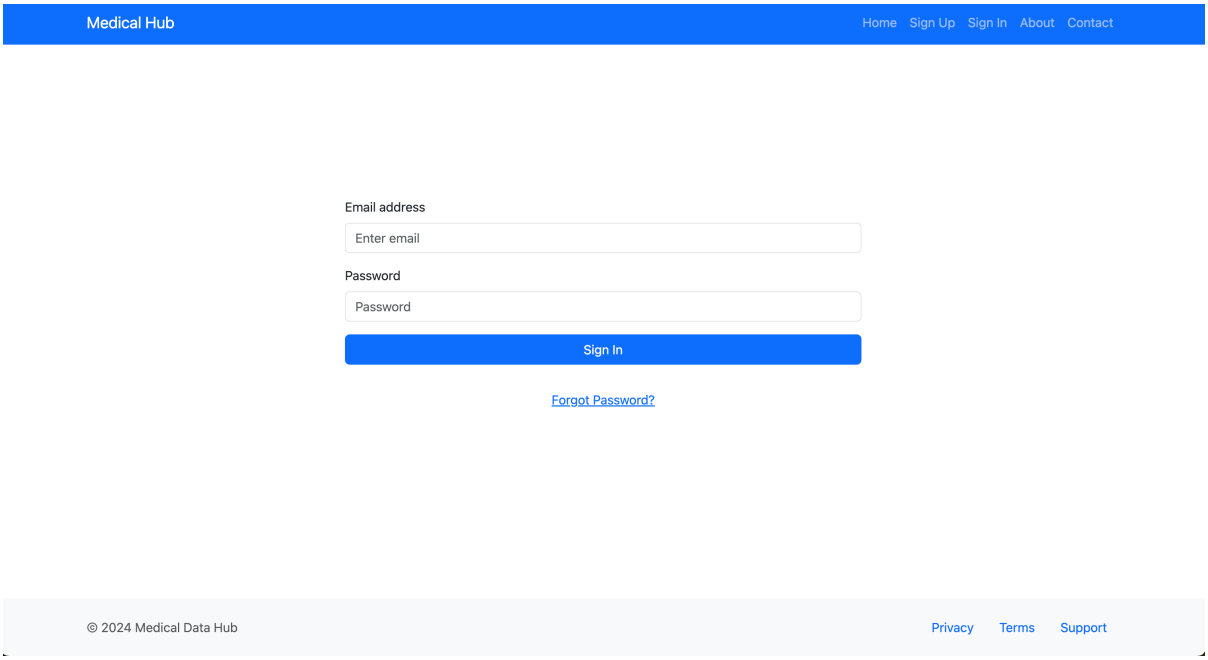

Figure 6.3: Medical Data Hub Sign In Page

are essential for researchers to perform data analysis and train machine learning models on the platform. Future versions of the platform will focus on implementing these features, as well other features that weren't planned on the initial version.

The platform still requires other improvements, which aren't related to the features, but are extremely important from a software engineering perspective. Some of these improvements include:

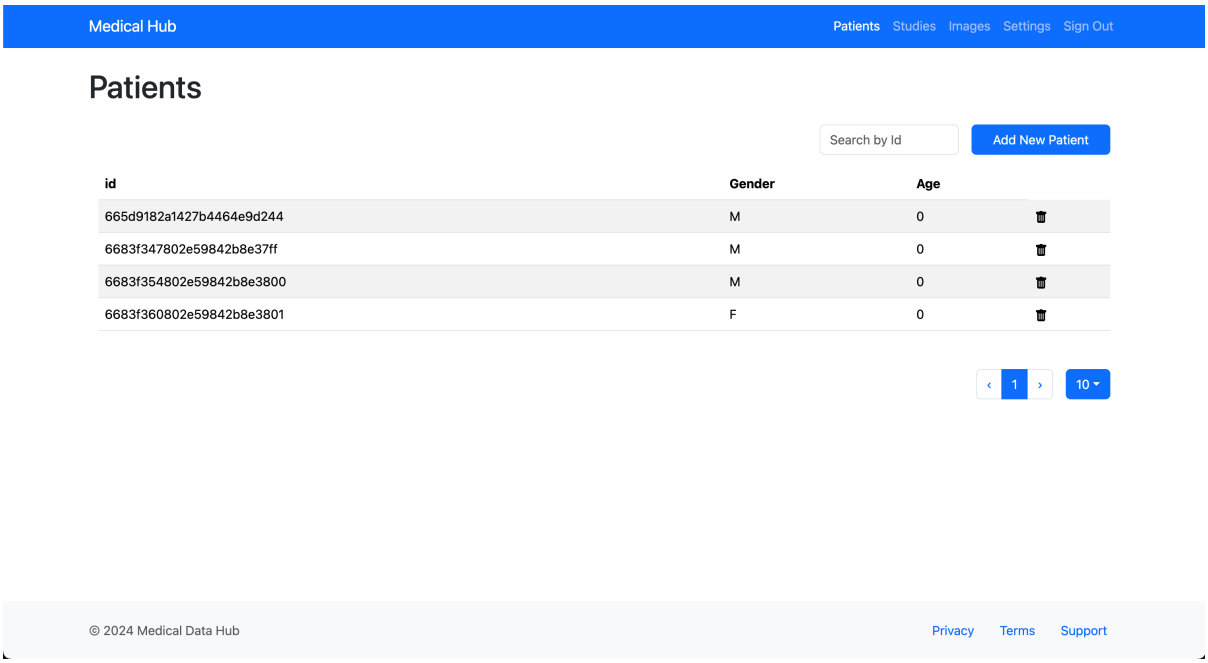

Figure 6.4: Medical Data Hub Patients Page

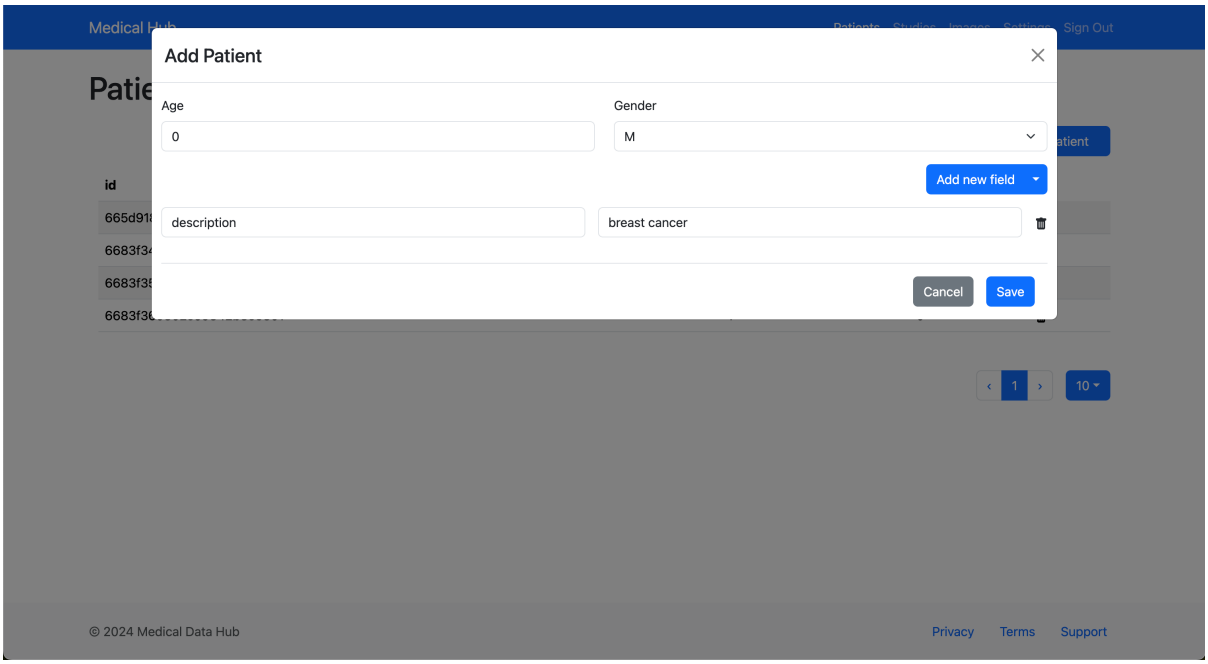

Figure 6.5: Medical Data Hub Add Patient Modal

- **Observability:** Implementing observability features, such as descriptive logging, monitoring, and tracing, to help developers debug and monitor the platform;
- **Deployment:** Implementing a Continuous Integration (CI)/Continuous Deployment (CD) pipeline to automate the deployment process and make it easier to deploy new versions of the platform;
- **Testing:** Implementing automated tests to ensure the platform's reliability and stability;

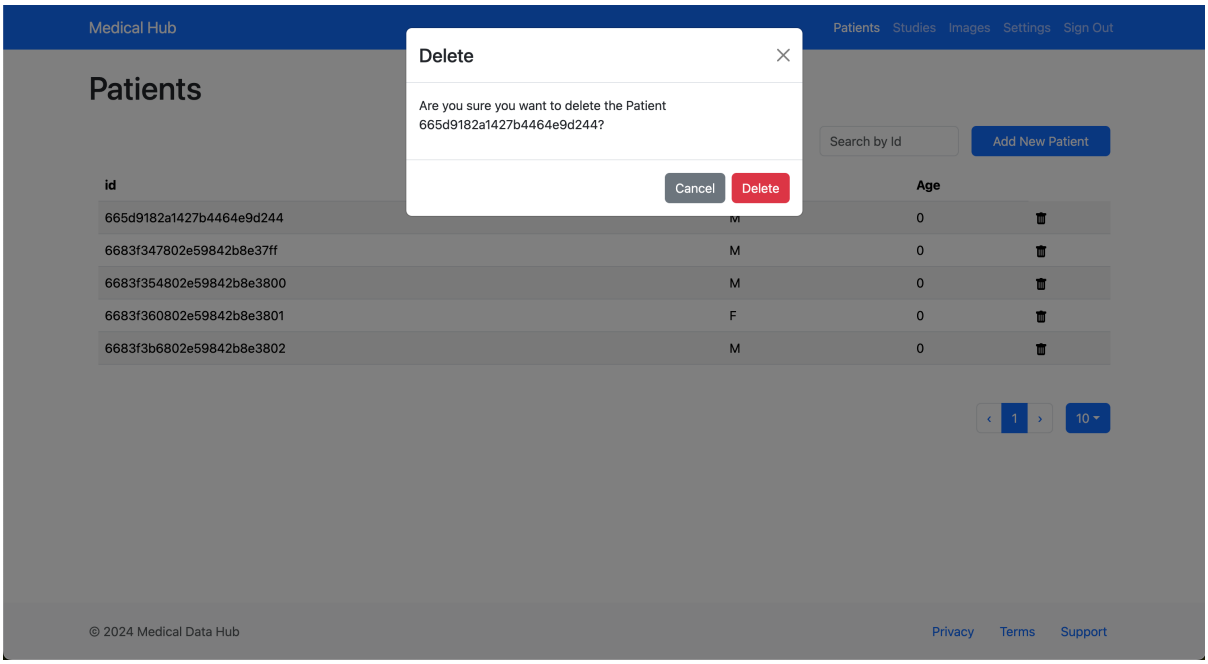

Figure 6.6: Medical Data Hub Delete Patient Modal

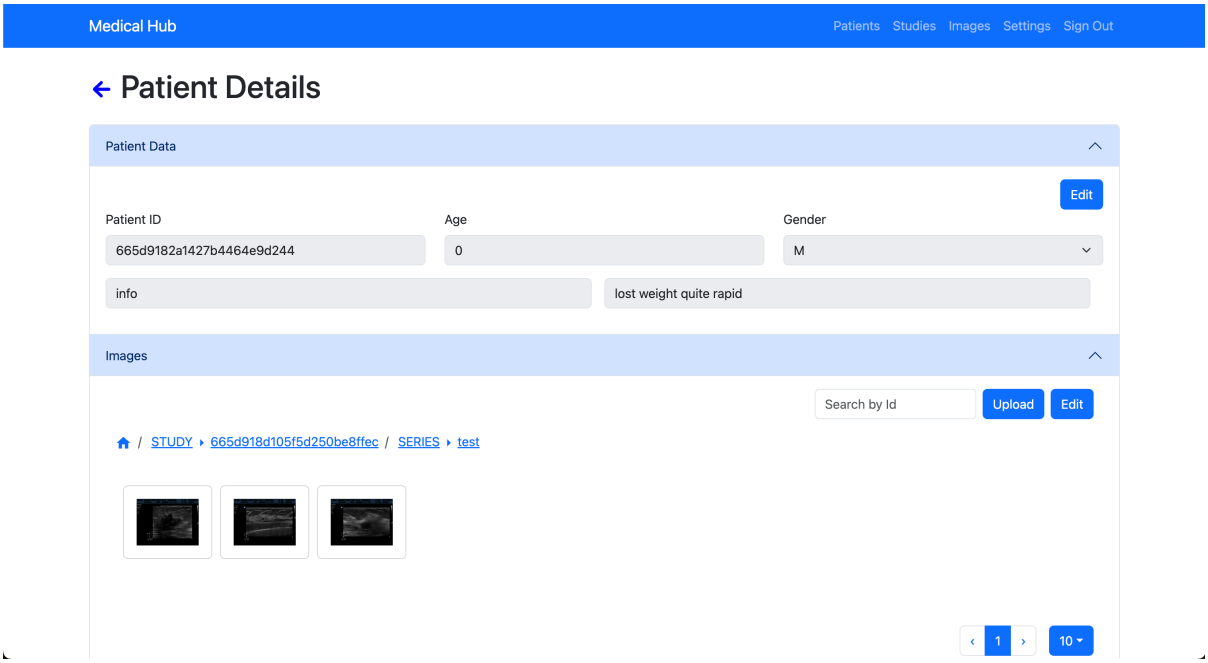

Figure 6.7: Medical Data Hub Patient Details Page

- **Backup and Recovery:** Implementing a backup and recovery strategy to ensure that data is not lost in case of a failure.

In terms of the decision of using microservices architecture, it was a good overall decision. However, the decoupling of the Patient and Study service could have had a better thought, both are interconnected and could have been bundle together in a single service. Gladly, the beauty of microservices is that allows for easy refactoring and changes, so if in the future

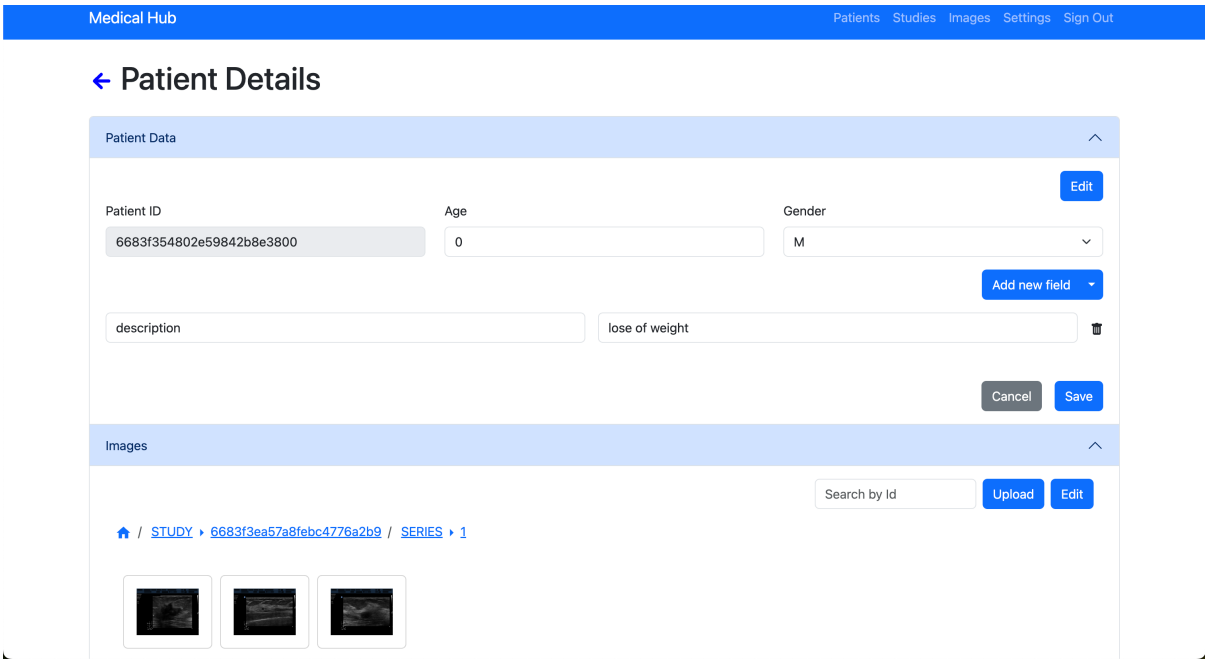

Figure 6.8: Medical Data Hub Patient Details Edit Mode

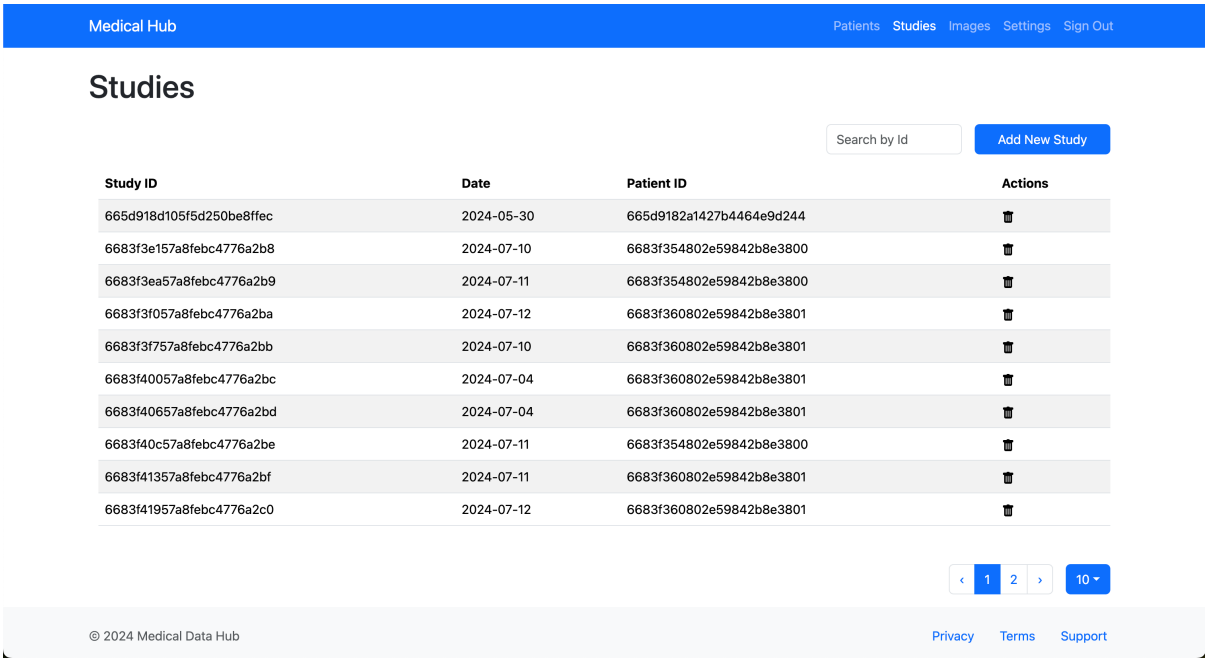

Figure 6.9: Medical Data Hub Studies Page

it is observed that both services need to be merged, it would be an easy relative task. I've mention in particular the previous situation, because when using microservices architecture, the tendency is to create more services than necessary, which can lead to a more complex system that needed.

Relative to the technologies used, they were relative a good choice as it solved and provided the necessary features for the platform. During the research phase, technologies that were

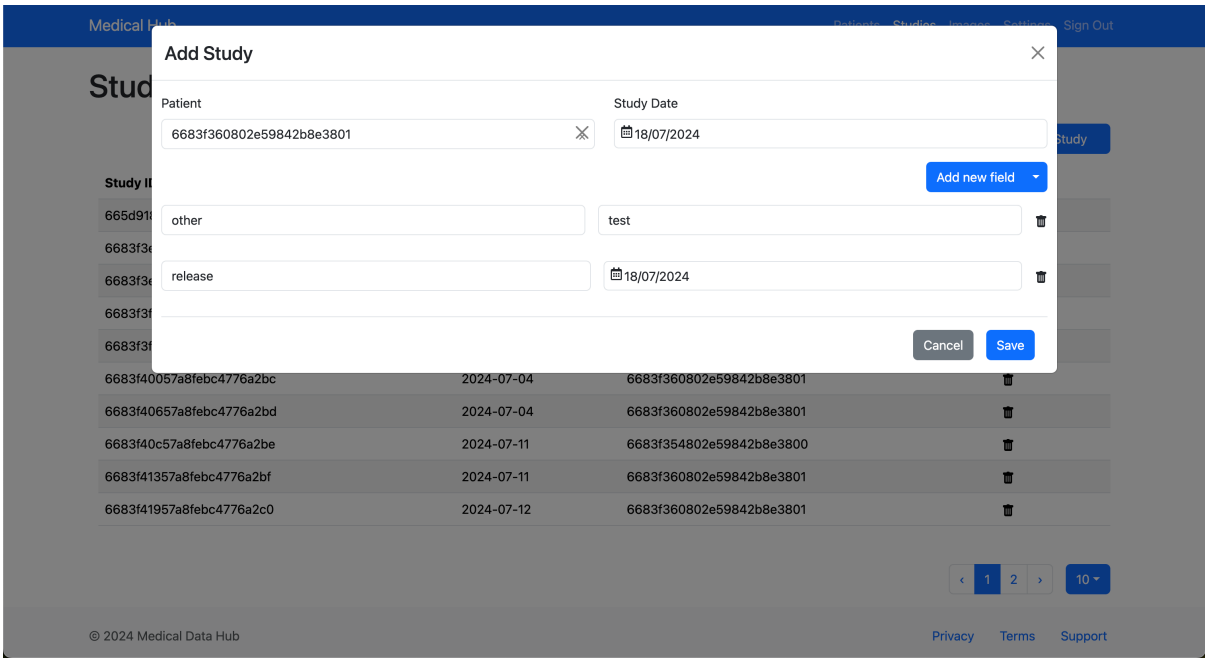

Figure 6.10: Medical Data Hub Add Study Modal

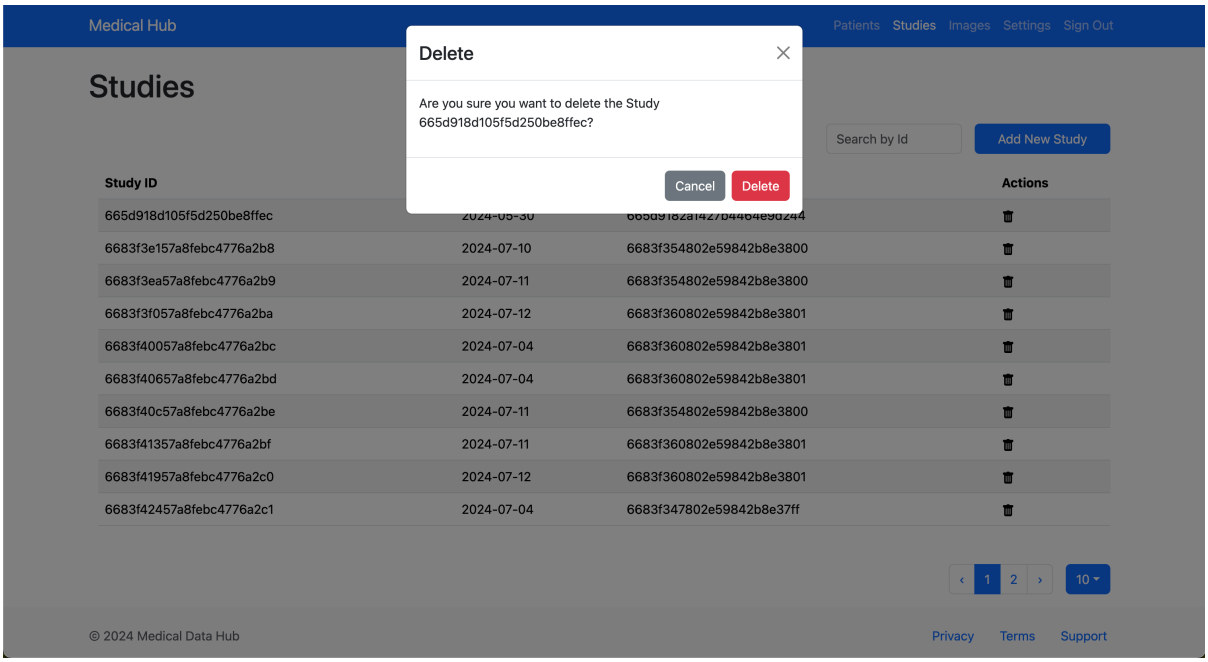

Figure 6.11: Medical Data Hub Delete Study Modal

more mature and had strong community support were selected. In retrospective, this decision proved beneficial, as it ensured good support and documentation, enabling to solve problems that were faced during the development phase.

The design of the platform was thought to be scalable and flexible, so it could be easily extended and improved in the future. The platform was designed to be modular, with each feature being implemented as a separate microservice, which allows for easy extension and

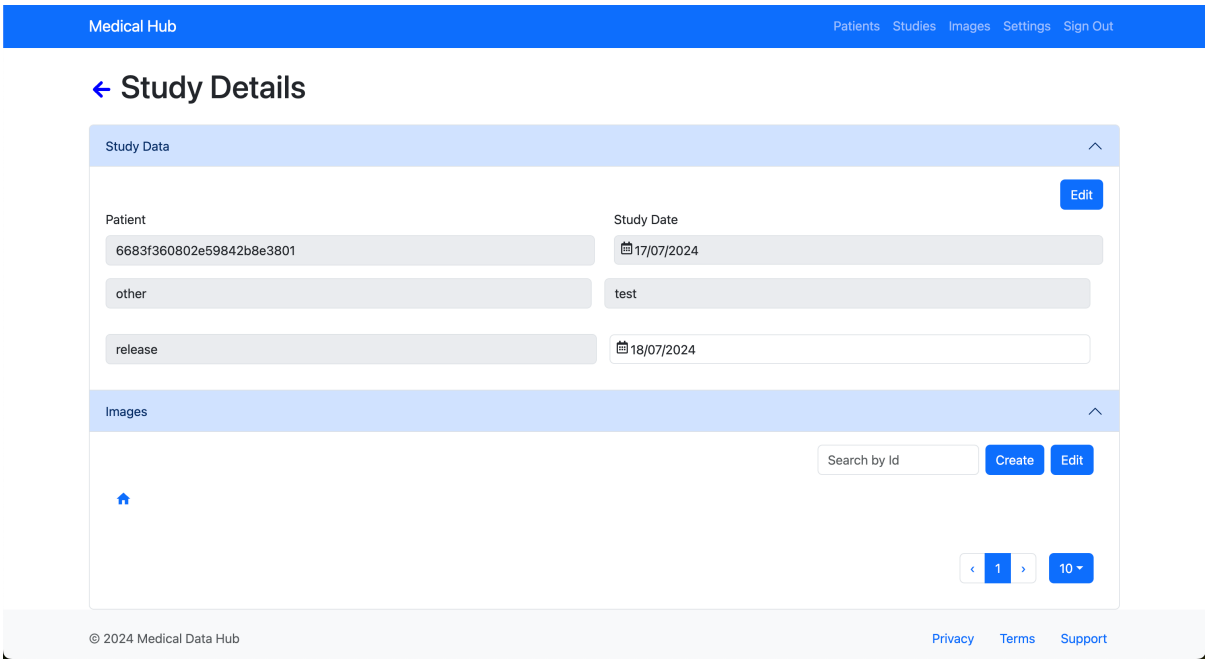

Figure 6.12: Medical Data Hub Study Details Page

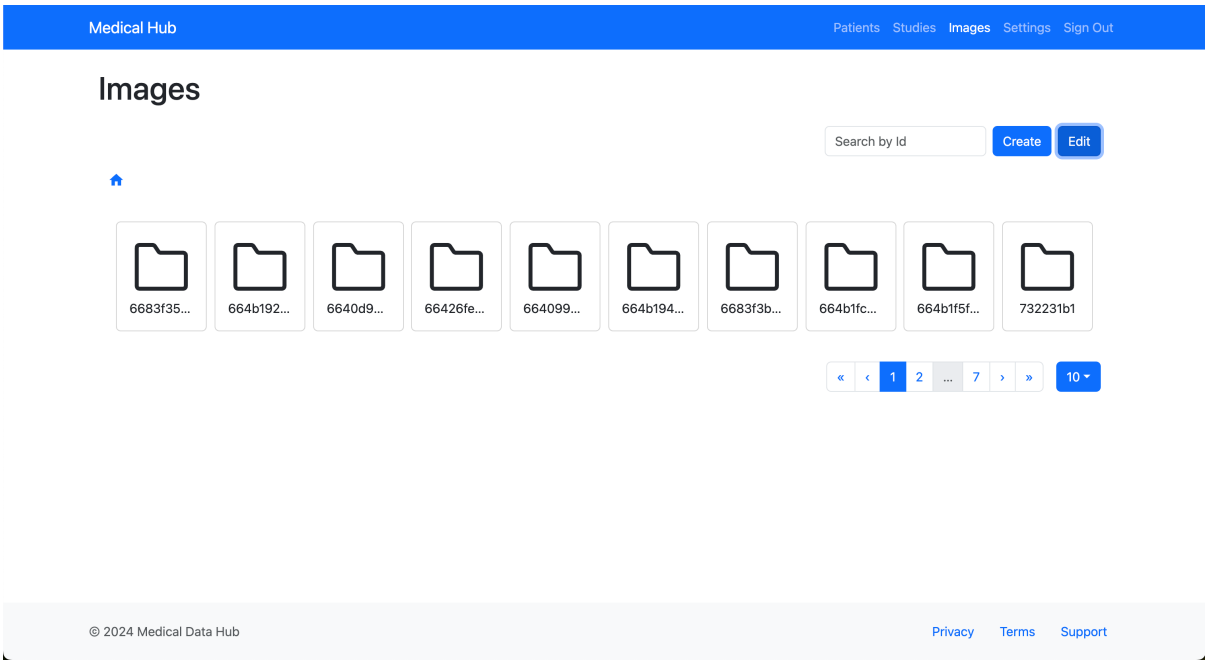

Figure 6.13: Medical Data Hub Images Page

modification of the platform. This design decision was made to ensure that the platform could be easily adapted to the needs of researchers and other users, as well as to ensure that the platform could be easily maintained and updated in the future. This will give an advantage in the future, as it will be easier to add new features and improve the platform without having to rewrite large parts of the codebase, which allows responding fast for the real-word requirements.

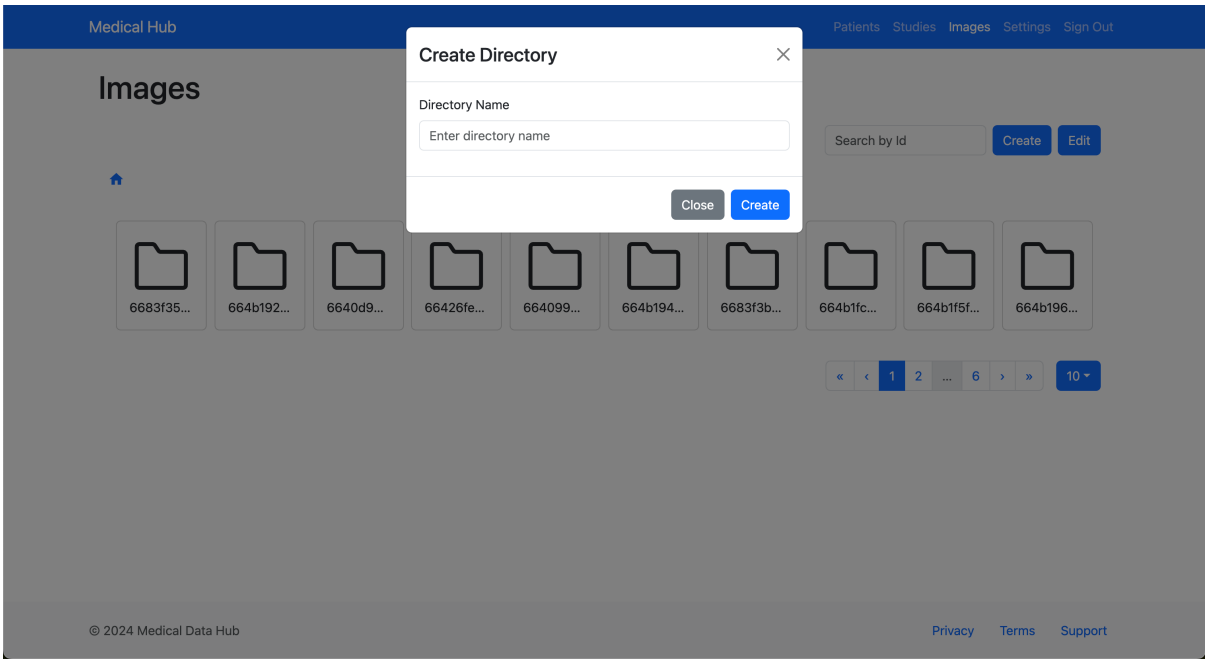

Figure 6.14: Medical Data Hub Create Directory Modal

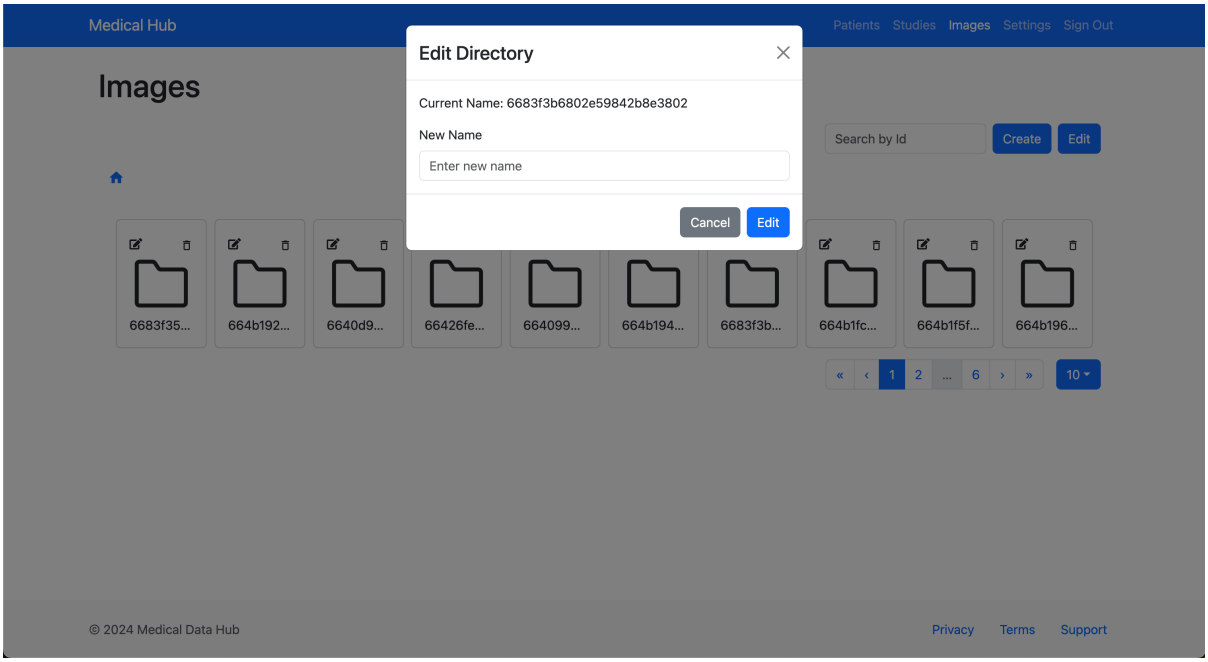

Figure 6.15: Medical Data Hub Edit Directory Name Modal

The authentication and authorization system was designed to delegate authorization decisions to the microservices. This approach allows for a more secure system, as authorization is centralized and controlled by the API Gateway. The decision was made based on the principle of the least privilege, which states that users should only have access to the resources they need to perform their tasks [89]. This principle was applied to the platform to ensure that users only have access to the data and features they need to perform their research. In retrospect,

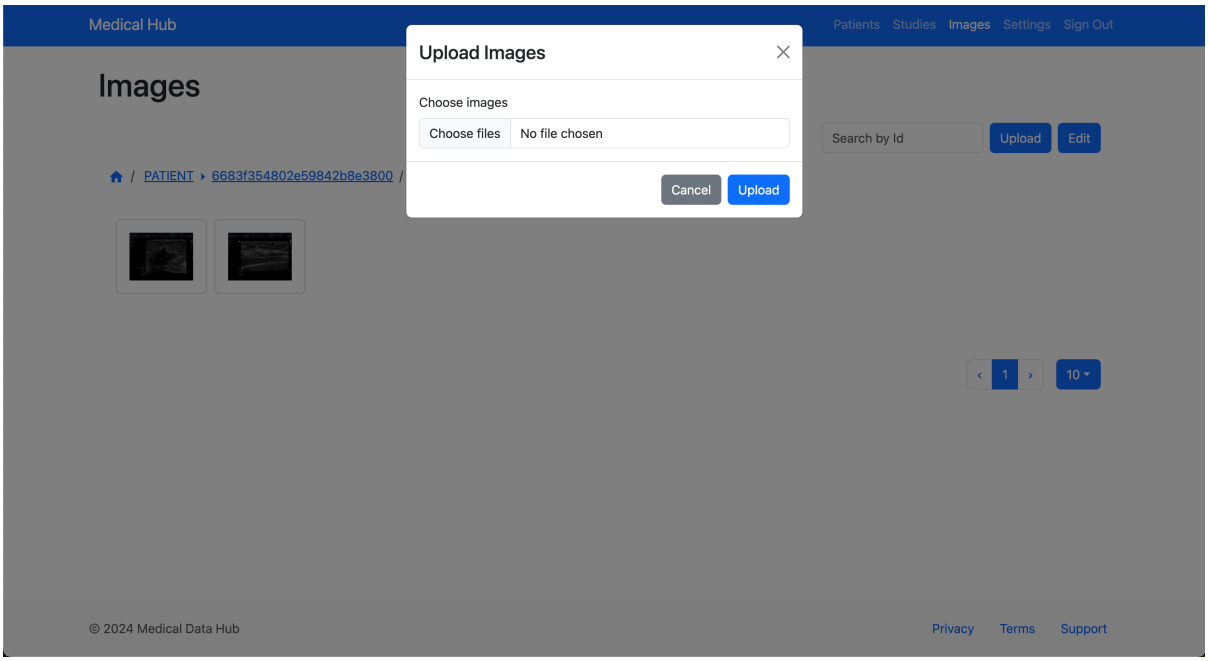

Figure 6.16: Medical Data Hub Upload Images Modal

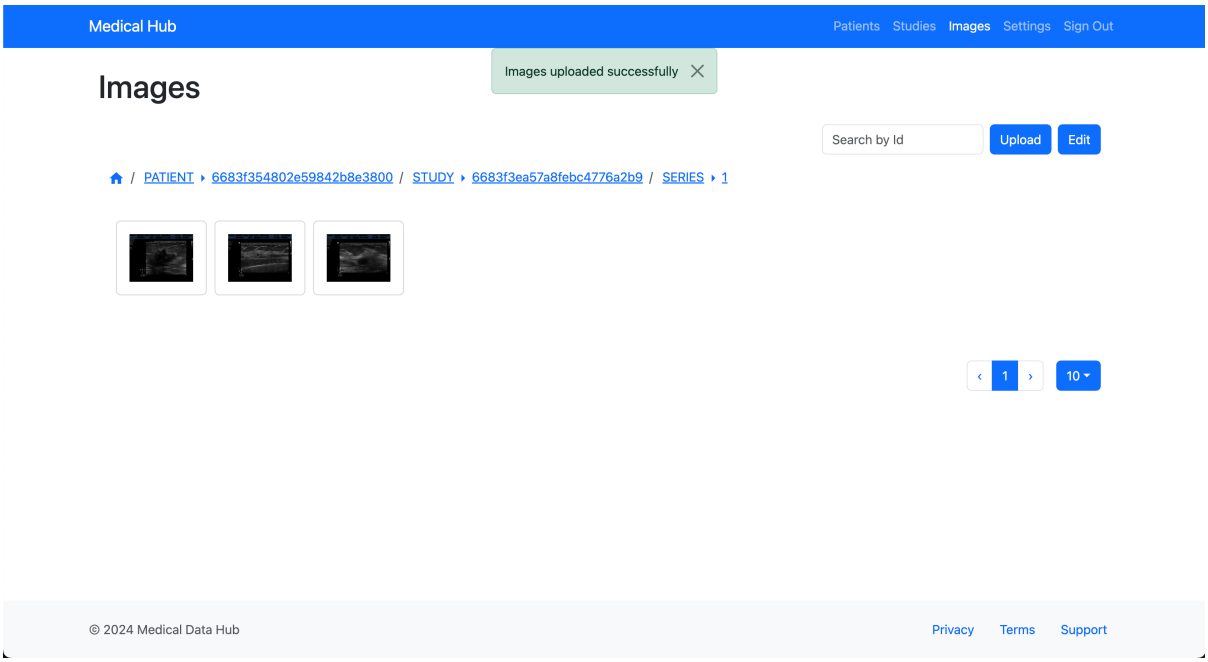

Figure 6.17: Medical Data Hub Image Uploaded Notification

the decision was a good one, as it allowed for a more secure system. However, this decision also adds some complexity to the system as we've to ensure that the authorization is correctly implemented in all microservices.

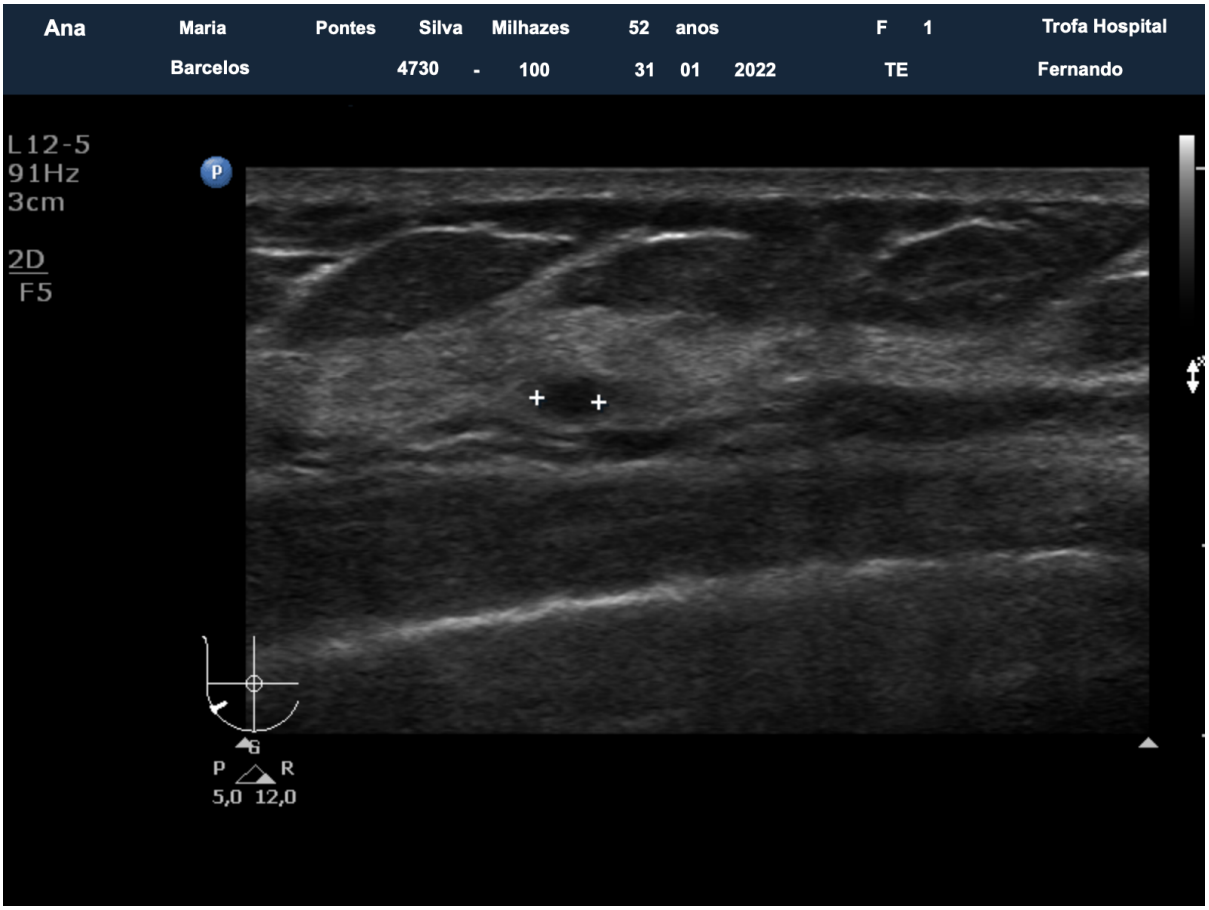

Figure 6.18: DICOM Image with Patient Information

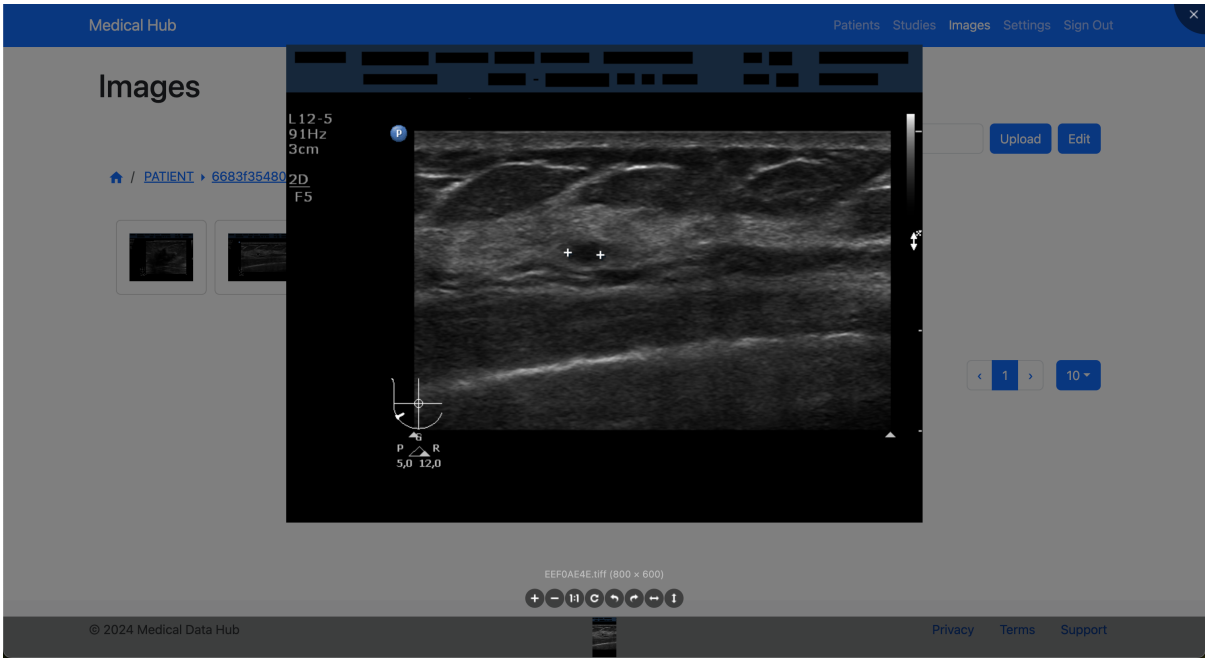

Figure 6.19: Medical Data Hub Image Viewer with Data Anonymized

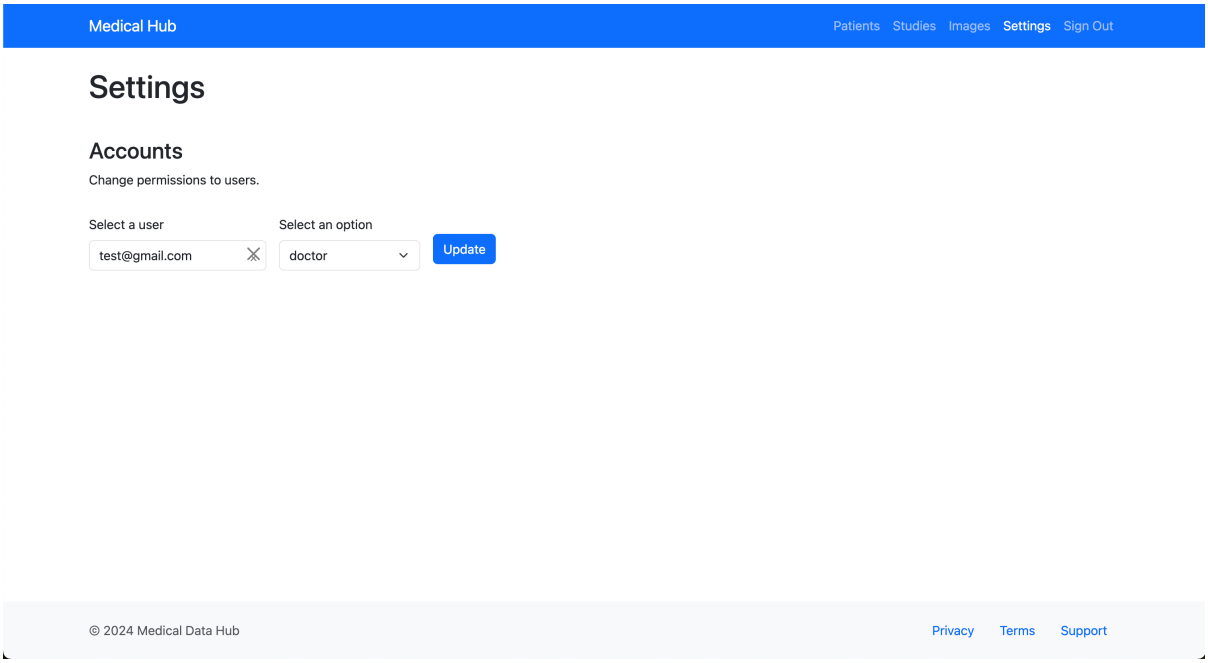

Figure 6.20: Medical Data Hub Settings Page

7 | Conclusion and Future Work

"Our imagination is the only limit to what we can hope to have in the future."

– Franklin D. Roosevelt

7.1 Conclusion

This dissertation presented a comprehensive solution to address the challenges posed by the new privacy regulations in accessing and utilizing medical data for research purposes. The developed platform, successfully provides a secure and anonymous environment for storing, accessing, and utilizing medical data, while ensuring compliance with GDPR regulations.

The platform's web-based graphical user interface allows researchers and medical professionals, to easily upload, explore, and analyze medical data, including DICOM images, patient information, and other types of data. Key features include user authentication and authorization, role-based access control, DICOM file handling, image anonymization, and data visualization tools. By addressing this critical gap in the field of medical data management and research, the platform provides a valuable tool for researchers and medical professionals to conduct research and develop new insights into diseases detection and treatment.

The involvement of researchers in the design process has ensured that the platform is not only user-friendly but also aligned with the real-world needs of its intended users. While there is still a lot of work to be done, the platform's current achievements lay a strong foundation for future development. The Medical Data Hub has the potential to significantly advance medical research by providing a robust tool for data management and machine learning model training, ultimately contributing to the early detection and treatment of diseases such as cancer.

7.2 Future Work

While the platform lays a strong foundation, several important features and improvements remain to be implemented. Some of the future work that could be done to enhance the platform includes:

- **Machine Learning Integration:** Implement a service to integrate machine learning models into the platform. This would enable the analysis of medical data and the development of predictive models. Technologies such as TensorFlow [90] and PyTorch [91] could be used.
- **Data Export:** Implement a service to export medical data in various formats, such as CSV or JSON. This would allow researchers to easily extract and analyze the data using external tools.
- **CI/CD Pipeline:** Set up a continuous integration and deployment pipeline to automate the build, testing, and deployment processes. This would ensure that changes to the platform are thoroughly tested and deployed reliably. Technologies such as GitHub Actions [92] and Jenkins [93] can be used.
- **Observability System:** Implement a monitoring and logging system to track the performance and health of the platform. This would help identify and resolve issues quickly, ensuring the platform remains stable and reliable. Technologies such as Datadog [94], Grafana [95], Prometheus [96], and OpenTelemetry [97] can be used.
- **Automatic Testing:** Develop automated tests to validate the functionality of the platform. This would help catch bugs and regressions early in the development process, improving the overall quality of the platform. Technologies such as Jest [98], Mocha [99], and Chai [100] can be used.
- **Automatic Backup System:** Implement a system to automatically back up the medical data stored in the platform. This would provide an additional layer of data protection and ensure that data can be restored in case of any unexpected events.
- **Stakeholder Feedback:** Discuss with medical professionals and researchers to understand their needs and requirements, and to gather further feedback on the platform. This would help identify areas for improvement and guide the development of new features.
- **Configuration Management:** Implement a configuration management feature that allows to configure the metadata to be extracted from the DICOM files and additionally

anonymize this data.

- **Read DICOMDIR:** Implement a feature to read the special DICOMDIR file and extract the information from it.
- **Export to DICOM:** Implement a feature to export the data to DICOM files.

By addressing these areas, the Medical Data Hub can continue to evolve and grow as a valuable tool in the medical research field, ultimately contributing to the development of technologies that can save lives through early disease detection and improved patient care.

In conclusion, the Medical Data Hub platform provides a significant contribution to the field of medical data management and research. By addressing key challenges such as data accessibility, security, and regulatory compliance, the platform provides a valuable tool for researchers and medical professionals. While there is still much work to be done, I am confident that the platform can be a valuable tool for the research community in the medical field. I sincerely hope that it can be used to improve the quality of life for many people who face the challenge of fighting diseases such as cancer.

Bibliography

[1] O.S. Pianykh. *Digital imaging and communications in medicine (DICOM): A practical introduction and survival guide*. 01 2008. doi: 10.1007/978-3-540-74571-6.

[2] Galen MRI. Galen mri, 2024. URL `https://www.galenmri.com/emma-dicom-images`. Accessed on 2024-06-23.

[3] Freddie Bray, Mathieu Laversanne, Hyuna Sung, Jacques Ferlay, Rebecca L. Siegel, Isabelle Soerjomataram, and Ahmedin Jemal. Global cancer statistics 2022: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries. *CA: A Cancer Journal for Clinicians*, 74(3):229–263, 2024. doi: https://doi.org/10.3322/caac.21834. URL `https://acsjournals.onlinelibrary.wiley.com/doi/abs/10.3322/caac.21834`.

[4] National Cancer Institute. Breast cancer screening (pdq), 2022. URL `https://www.cancer.gov/types/breast/patient/breast-screening-pdq`. Accessed on 2024-06-24.

[5] Solveig Hofvind, Per Skaane, Bedrich Vitak, and et al. Influence of review design on percentages of missed interval breast cancers: retrospective study of interval cancers in a population-based screening program. *Radiology*, 237(2):437–443, 2005. doi: 10.1148/radiol.2372041174.

[6] Md. Milon Islam, Md. Rezwanul Haque, Hasib Iqbal, Md. Munirul Hasan, Mahmudul Hasan, and Muhammad Nomani Kabir. Breast cancer prediction: A comparative study using machine learning techniques. *SN Computer Science*, 1(5):290, 2020. doi: 10.1007/s42979-020-00305-w. URL `https://doi.org/10.1007/s42979-020-00305-w`.

[7] Mohammed Amine Naji, Sanaa El Filali, Kawtar Aarika, EL Habib Benlahmar, Rachida Ait Abdelouhahid, and Olivier Debauche. Machine learning algorithms for breast cancer prediction and diagnosis. *Procedia Computer Science*, 191:487–492, 2021.

ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2021.07.062. URL https://www.sciencedirect.com/science/article/pii/S1877050921014629. The 18th International Conference on Mobile Systems and Pervasive Computing (MobiSPC), The 16th International Conference on Future Networks and Communications (FNC), The 11th International Conference on Sustainable Energy Information Technology.

[8] Heidi Beate Bentzen, Rosa Castro, Robin Fears, George Griffin, Volker ter Meulen, and Giske Ursin. Remove obstacles to sharing health data with researchers outside of the european union. *Nature Medicine*, 27(8):1329–1333, 2021. doi: 10.1038/s41591-021-01460-0. URL https://doi.org/10.1038/s41591-021-01460-0.

[9] Oya Beyan, Ananya Choudhury, Johan van Soest, Oliver Kohlbacher, Lukas Zimmermann, Holger Stenzhorn, Md. Rezaul Karim, Michel Dumontier, Stefan Decker, Luiz Olavo Bonino da Silva Santos, and Andre Dekker. Distributed Analytics on Sensitive Medical Data: The Personal Health Train. *Data Intelligence*, 2(1-2):96–107, 01 2020. ISSN 2641-435X. doi: 10.1162/dint_a_00032. URL https://doi.org/10.1162/dint_a_00032.

[10] Tilburg University. Assessment of the eu member states' rules on health data in the light of gdpr, 2021. URL https://research.tilburguniversity.edu/en/publications/assessment-of-the-eu-member-states-rules-on-health-data-in-the-li. Accessed on 2024-07-25.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[12] THE EUROPEAN PARLIAMENT and THE COUNCIL OF THE EUROPEAN UNION. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016. *Official Journal of the European Union*, 2016.

[13] European Union. Eur-lex, 2021. URL https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32021R2282. Accessed on 2024-04-24.

[14] THE EUROPEAN PARLIAMENT and THE COUNCIL OF THE EUROPEAN UNION. Principles relating to processing of personal data. *Official Journal of the European Union*, page Article 5, 2016.

[15] W D Bidgood and S C Horii. Introduction to the acr-nema dicom standard. *Radio-*

Graphics, 12(2):345–355, 1992. doi: 10.1148/radiographics.12.2.1561424. URL https://doi.org/10.1148/radiographics.12.2.1561424. PMID: 1561424.

[16] Dicom standard, 2024. URL https://dicom.nema.org/medical/dicom/current/output/html/part01.html#chapter_Notice. Accessed on 2024-05-01.

[17] J. F. Kurose and K. W. Ross. *Computer Networking: A Top-Down Approach*. Pearson, 7th edition, 2017.

[18] Mahadevappa Mahesh. The essential physics of medical imaging, third edition. *Medical Physics*, 2013. doi: https://doi.org/10.1118/1.4811156. URL https://aapm.onlinelibrary.wiley.com/doi/abs/10.1118/1.4811156.

[19] European Commission. European cancer information system (ecis). *Knowledge4Policy*, 2022. URL https://knowledge4policy.ec.europa.eu/cancer/topic/cancer-information/ecis_en. Accessed on 2024-05-28.

[20] Dicom library, 2011. URL https://www.dicomlibrary.com/. Accessed on 2024-06-17.

[21] Kenneth Clark, Bruce Vendt, Kirk Smith, John Freymann, Justin Kirby, Paul Koppel, Stephen Moore, Stanley Phillips, Derek Maffitt, Michael Pringle, Lawrence Tarbox, and Fred Prior. The cancer imaging archive (tcia): Maintaining and operating a public information repository. *Journal of Digital Imaging*, 26(6):1045–1057, 2013. ISSN 1618-727X. doi: 10.1007/s10278-013-9622-7. URL https://doi.org/10.1007/s10278-013-9622-7.

[22] UK Biobank. Uk biobank. *UK Biobank*, 2022. URL https://www.ukbiobank.ac.uk/. Accessed on 2024-05-28.

[23] Genomic Data Commons. Gdc portal, 2024. URL https://portal.gdc.cancer.gov/. Accessed on 2024-05-28.

[24] Miguel G Lopez, Naimy Posada, Daniel C Moura, Raúl Ramos Pollán, José M Franco Valiente, César Suárez Ortega, Manuel Solar, Guillermo Diaz-Herrero, IMAP Ramos, Joana Loureiro, Teresa Cardoso Fernandes, and BM Ferreira de Araujo. Bcdr: a breast cancer digital repository. 1215:113–120, 2012. URL https://www.researchgate.net/publication/258243150_BCDR_A_BREAST_CANCER_DIGITAL_REPOSITORY.

[25] Scott Tilley. *Systems Analysis and Design*. 2019.

[26] Excalidraw. Excalidraw, 2024. URL https://docs.excalidraw.com/. Accessed on 2024-07-25.

[27] MongoDB. Mongodb, 2024. URL https://www.mongodb.com/. Accessed on 2024-07-25.

[28] Atlassian. Jira, 2024. URL https://www.atlassian.com/software/jira. Accessed on 2024-07-31.

[29] Git. Git, 2024. URL https://git-scm.com/. Accessed on 2024-07-31.

[30] GitHub. Github, 2024. URL https://github.com/. Accessed on 2024-07-31.

[31] Microsoft. Visual studio code, 2024. URL https://code.visualstudio.com/. Accessed on 2024-07-31.

[32] Docker. Docker, 2024. URL https://www.docker.com/. Accessed on 2024-07-26.

[33] Next.js. Next.js, 2024. URL https://nextjs.org/. Accessed on 2024-07-31.

[34] React. React, 2024. URL https://react.dev/. Accessed on 2024-07-31.

[35] TypeScript. Typescript, 2024. URL https://www.typescriptlang.org/. Accessed on 2024-07-31.

[36] React Bootstrap. React bootstrap, 2024. URL https://react-bootstrap.netlify.app/. Accessed on 2024-07-31.

[37] Node.js. Node.js, 2024. URL https://nodejs.org/. Accessed on 2024-07-31.

[38] Express.js. Express.js, 2024. URL https://expressjs.com/. Accessed on 2024-07-31.

[39] Python Software Foundation. Python, 2024. URL https://www.python.org/. Accessed on 2024-07-31.

[40] Sebastián Ramírez. Fastapi, 2024. URL https://fastapi.tiangolo.com/. Accessed on 2024-07-31.

[41] W3Schools. Html css tutorial, 2024. URL https://www.w3schools.com/html/html_css.asp. Accessed on 2024-07-31.

[42] React Grid Layout. React grid layout, 2024. URL https://github.com/react-grid-layout/react-grid-layout. Accessed on 2024-07-31.

[43] React Datepicker. React datepicker, 2024. URL https://reactdatepicker.com/. Accessed on 2024-07-31.

[44] Eric Giovangigli. React bootstrap typeahead, 2024. URL https://github.com/ericgio/react-bootstrap-typeahead. Accessed on 2024-07-31.

[45] ESLint. Eslint, 2024. URL https://eslint.org/. Accessed on 2024-07-31.

[46] JavaScript. Javascript, 2024. URL https://www.javascript.com/. Accessed on 2024-07-31.

[47] MongoDB. Pymongo, 2024. URL https://pymongo.readthedocs.io/en/stable/. Accessed on 2024-07-31.

[48] ldapjs. ldapjs, 2024. URL http://ldapjs.org/. Accessed on 2024-07-31.

[49] npm. http-proxy-middleware, 2024. URL https://www.npmjs.com/package/http-proxy-middleware. Accessed on 2024-07-31.

[50] pydicom. pydicom, 2024. URL https://pydicom.github.io/. Accessed on 2024-07-31.

[51] Microsoft. Presidio, 2024. URL https://microsoft.github.io/presidio/. Accessed on 2024-07-31.

[52] npm. dotenv, 2024. URL https://www.npmjs.com/package/dotenv. Accessed on 2024-07-31.

[53] npm. Cors, 2024. URL https://www.npmjs.com/package/cors. Accessed on 2024-07-31.

[54] npm. body-parser, 2024. URL https://www.npmjs.com/package/body-parser. Accessed on 2024-07-31.

[55] npm. cookie, 2024. URL https://www.npmjs.com/package/cookie. Accessed on 2024-07-31.

[56] npm. jsonwebtoken, 2024. URL https://www.npmjs.com/package/jsonwebtoken. Accessed on 2024-07-31.

[57] npm. morgan, 2024. URL https://www.npmjs.com/package/morgan. Accessed on 2024-07-31.

[58] npm. swagger-ui-express, 2024. URL https://www.npmjs.com/package/swagger-ui-express. Accessed on 2024-07-31.

[59] npm. swagger-jsdoc, 2024. URL https://www.npmjs.com/package/swagger-jsdoc. Accessed on 2024-07-31.

[60] pydantic. pydantic, 2024. URL https://docs.pydantic.dev/latest/. Accessed on 2024-07-31.

[61] uvicorn. uvicorn, 2024. URL https://www.uvicorn.org/. Accessed on 2024-07-31.

[62] numpy. numpy, 2024. URL https://numpy.org/. Accessed on 2024-07-31.

[63] pillow. pillow, 2024. URL https://pypi.org/project/pillow/. Accessed on 2024-07-31.

[64] python-multipart. python-multipart, 2024. URL https://pypi.org/project/python-multipart/. Accessed on 2024-07-31.

[65] LDAP.com. Ldap.com, 2024. URL https://ldap.com/. Accessed on 2024-07-31.

[66] Wikipedia. File system, 2024. URL https://en.wikipedia.org/wiki/File_system. Accessed on 2024-07-31.

[67] jwt.io. jwt.io, 2024. URL https://jwt.io/. Accessed on 2024-07-31.

[68] PyJWT. Pyjwt documentation, 2024. URL https://pyjwt.readthedocs.io/en/stable/. Accessed on 2024-07-31.

[69] npm. bcrypt, 2024. URL https://www.npmjs.com/package/bcrypt. Accessed on 2024-07-31.

[70] GitKraken. Gitkraken, 2024. URL https://www.gitkraken.com/. Accessed on 2024-07-31.

[71] OrbStack. Orbstack, 2024. URL https://orbstack.dev/. Accessed on 2024-07-31.

[72] Postman. Postman, 2024. URL https://www.postman.com/. Accessed on 2024-07-31.

[73] Atlassian. Confluence, 2024. URL https://www.atlassian.com/software/confluence. Accessed on 2024-07-31.

[74] PyPI. pip, 2024. URL https://pypi.org/project/pip/. Accessed on 2024-07-31.

[75] npm. npm, 2024. URL https://www.npmjs.com/. Accessed on 2024-07-31.

[76] leenooks. phpldapadmin, 2024. URL https://github.com/leenooks/phpLDAPadmin. Accessed on 2024-07-31.

[77] mongo-express. mongo-express, 2024. URL https://github.com/mongo-express/mongo-express. Accessed on 2024-07-31.

[78] npm. Http status codes, 2024. URL https://www.npmjs.com/package/http-status-codes. Accessed on 2024-07-31.

[79] Angular. Angular, 2024. URL https://angular.dev/. Accessed on 2024-07-31.

[80] Vue.js. Vue.js, 2024. URL https://vuejs.org/. Accessed on 2024-07-31.

[81] Pallets. Flask, 2024. URL https://flask.palletsprojects.com/en/3.0.x/. Accessed on 2024-07-31.

[82] Django Software Foundation. Django, 2024. URL https://www.djangoproject.com/. Accessed on 2024-07-31.

[83] Ruby. Ruby, 2024. URL https://www.ruby-lang.org/en/. Accessed on 2024-07-31.

[84] Oracle. Java, 2024. URL https://www.java.com/en/. Accessed on 2024-07-31.

[85] MySQL. Mysql, 2024. URL https://www.mysql.com/. Accessed on 2024-07-31.

[86] OAuth. Oauth 2.0, Year. URL https://oauth.net/2/. Accessed on 2024-07-31.

[87] Creativyst. Csv file format tutorial, 2024. URL https://www.creativyst.com/Doc/Articles/CSV/CSV01.shtml. Accessed on 2024-07-31.

[88] JSON. Json, 2024. URL https://www.json.org/json-en.html. Accessed on 2024-07-31.

[89] J.H. Saltzer and M.D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975. doi: 10.1109/PROC.1975.9939.

[90] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal

Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[91] PyTorch. Pytorch, 2024. URL `https://pytorch.org/`. Accessed on 2024-07-31.

[92] GitHub. Github actions, 2024. URL `https://github.com/features/actions`. Accessed on 2024-07-31.

[93] Jenkins. Jenkins, 2024. URL `https://www.jenkins.io/`. Accessed on 2024-07-31.

[94] Datadog. Datadog, 2024. URL `https://www.datadoghq.com/`. Accessed on 2024-07-31.

[95] Grafana Labs. Grafana, 2024. URL `https://grafana.com/`. Accessed on 2024-07-31.

[96] Prometheus. Prometheus, 2024. URL `https://prometheus.io/`. Accessed on 2024-07-31.

[97] OpenTelemetry. Opentelemetry, 2024. URL `https://opentelemetry.io/`. Accessed on 2024-07-31.

[98] Jest. Jest, 2024. URL `https://jestjs.io/`. Accessed on 2024-07-31.

[99] Mocha. Mocha, 2024. URL `https://mochajs.org/`. Accessed on 2024-07-31.

[100] Chai. Chai, 2024. URL `https://chaijs.com/`. Accessed on 2024-07-31.