



Heinrich Heine University Düsseldorf

Institute for Computer Science

Lehrstuhl Prof. Dr. Heider

**Abschlussarbeit zur Erlangung des akademischen Grades Bachelor of Science
im Studiengang Informatik**

Bachelorarbeit von

Mohammad Sharabati

Documentation-Based Software Development for Skin Cancer Detection

Sommersemester 2024

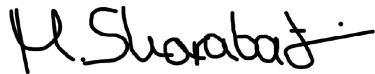
Date of submission: 16.04.2024

Reviewers: Prof. Dr. Dominik Heider
Prof. Dr. Gunnar Klau

Declaration of Independence

I, **Mohammad Sharabati**, hereby declare that I have written this thesis independently and have not used any sources and aids other than those specified. All statements taken from other sources, either literally or analogously, are labeled as such.

Düsseldorf, 16.04.2024

A handwritten signature in black ink, appearing to read 'M. Sharabati', with a horizontal line extending from the end.

(Mohammad Sharabati)

Abstract

Skin cancer is among the most common types of cancer. To aid medical professionals in diagnosing and treating various types of skin cancer, a range of devices and software, including dermatoscopes and digital imaging systems, are utilized. However, the development of such software and its integration into the medical industry can be challenging for academic institutions and researchers due to strict regulations that are often perceived as unfeasible, complex, and time-consuming. This thesis aims to implement the development and integration of medical software into the healthcare sector, focusing on skin cancer detection, by developing a clinical decision support system that could assist physicians in the diagnosis and classification of skin lesions.

The methodology is based on the adoption and integration of four academia-tailored guidelines, namely the Quality Management System, Software Life Cycle, Risk Management, and Usability Engineering guidelines, into the development process of the machine learning software for the classification of skin lesions. This process is meticulously documented, from software specification through the coding, testing, and verification phases.

In this project, a software which can calculate skin cancer risk scores in a matter of seconds, based on a captured photograph, was developed. By proving that a structured, guideline-based approach can lead to the efficient development of medical software closer to regulatory standards, this project provides valuable insights into the field of medical informatics. It emphasizes the importance of clearly documented processes in overcoming challenges associated with regulatory approval and clinical adoption of medical software.

Zusammenfassung

Hautkrebs ist eine der häufigsten Krebsarten. Um Medizinern bei der Diagnose und Behandlung von Hautkrebsarten zu unterstützen, werden verschiedene Geräte und Software, wie Dermatoskope und digitale Bildgebungssysteme, eingesetzt. Die Entwicklung solcher Software und deren Integration in die medizinische Industrie stellt jedoch für akademische Institutionen und Forschende eine Herausforderung dar, da sie strengen Regulierungen unterliegt, die oft als unpraktikabel, komplex und zeitintensiv empfunden werden. Diese Arbeit zielt darauf ab, die Entwicklung und Integration medizinischer Software in den Gesundheitssektor zu vereinfachen, mit Fokus auf Erkennung von Hautkrebs, indem ein klinisches Entscheidungsunterstützungssystem entwickelt wird, das Ärzte bei der Diagnose und Klassifikation von Hautläsionen unterstützen könnte.

Die Methodologie basiert auf der Annahme und Integration von vier Hauptleitlinien, nämlich der Qualitätsmanagementsystem, Softwarelebenszyklus, Risikomanagement und Usability Engineering Leitlinien, in den Entwicklungsprozess einer Machine-Learning Software für die Klassifikation von Hautläsionen. Dieser Prozess wird sorgfältig dokumentiert, von der Spezifikation der Software bis zu den Phasen des Coding, Testing und Verifizierens.

In diesem Projekt wurde demnach eine Software entwickelt, die Krebsrisikowerte anhand eines Fotos einer Hautläsion in sekundenschnelle berechnen kann. Indem demonstriert wird, dass ein strukturierter, richtlinienbasierter Ansatz zur effizienten Entwicklung medizinischer Software führen kann, die näher an regulatorischer Konformität ist, bietet dieses Projekt wertvolle Erkenntnisse für das Feld der medizinischen Informatik. Es unterstreicht die Wichtigkeit klar dokumentierter Prozesse bei der Überwindung der Herausforderungen im Zusammenhang mit der regulatorischen Zulassung und der klinischen Einführung medizinischer Software.

Contents

1. Introduction	1
2. Background	3
2.1. Skin Lesions and Diagnosis	3
2.2. Machine Learning	5
2.2.1. Deep Learning and Convolutional Neural Networks	6
2.3. Medical Software	8
2.3.1. Medical Device Software	9
2.3.2. Software as a Medical Device	9
2.4. International Standards for Medical Software	9
2.4.1. The Medical Device Regulation	9
2.4.2. The In Vitro Diagnostic Regulation	10
2.4.3. Quality Management System	11
2.4.4. Software Life Cycle	12
2.4.5. Risk Management	14
2.4.6. Usability Engineering	14
2.5. Academia-tailored Guidelines for Medical Software	15
3. Materials and Methodology	16
3.1. Software Specification	17
3.1.1. Python Libraries and Packages	17
3.1.2. Qt Designer	17
3.1.3. Documentation Platforms	18
3.2. Machine Learning Pipeline	18
3.2.1. Skin Lesion Dataset: HAM10000	19
3.2.2. Preprocessing	19
3.2.3. Transfer Learning	19
3.2.4. Training Process	20
3.2.5. Chosen Model: DenseNet201	20
3.3. Guidelines for Software Development	21
3.3.1. Quality Management System Guideline	21
3.3.2. Software Life Cycle Guideline	23
3.3.3. Risk Management Guideline	25
3.3.4. Usability Engineering Guideline	28
3.4. Application and Implementation of the Four Guidelines	31
3.5. Software Implementation	31
3.5.1. Coding	31
3.5.2. Testing	31

4. Results	33
4.1. Initial Preparation	33
4.1.1. Setups	33
4.2. First Documentation Stage	34
4.2.1. Quality Management System Activities	34
4.2.2. Software Life Cycle Introduction	35
4.2.3. Intended Purpose	35
4.2.4. Safety Class Determination	35
4.2.5. Software Development Planning	36
4.2.6. Use Specification	37
4.2.7. Iterative Design Cycle Initialization	38
4.2.8. Risk Management Planning and Risk Policy	38
4.3. Second Documentation Stage	39
4.3.1. System and Software Requirements Specification	39
4.3.2. System and Software Documents	40
4.3.3. Iterative Design Cycle Continuation	41
4.3.4. Mockup and Class Diagram	41
4.4. Final Documentation Stage	42
4.4.1. Use-related Risk Analysis	42
4.4.2. Summative Evaluation	42
4.4.3. Iterative Risk Management Cycle	43
4.4.4. Verification and Release	47
4.5. Software Implementation	48
4.5.1. Coding	48
4.5.2. Testing	52
4.6. Final Software Design and Documents	54
5. Discussion	57
Bibliography	I

A. Appendix	VIII
A.1. Quality Management System Project Execution Extract	IX
A.2. Risk Policy	XI
A.3. Risk Table	XII
A.4. Jira Project Plan	XIII
A.5. Confluence Overview	XIV
A.6. Risk Control Measures	XV
A.7. Document Management	XVI
A.8. Quality Management System Project Planning Extract	XVIII
A.9. Intended Purpose	XIX
A.10.Safety Class Determination	XX
A.11.Software Development Plan	XXII
A.12.Use Specification	XXIV
A.13.Risk Management Preparation and Planning	XXV
A.14.System and Software Requirements Specification	XXVI
A.15.System and Software Design	XXX
A.16.Software Architecture Description	XXXII
A.17.Configuration and Change Management	XXXVII
A.18.Iterative Design Cycle 3 and 4	XXXVIII
A.19.Mockup Process	XLVIII
A.20.Class Diagram	XLIX
A.21.Use-related Risk Analysis	L
A.22.Summative Evaluation	LII
A.23.Risk Monitoring	LV
A.24.README.txt	LVII
A.25.Software Problem-solving Process	LVIII
A.26.Confluence Bibliography	LIX

List of Abbreviations

AI	Artificial Intelligence
AKIEC	Actinic Keratosis
ANN	Artificial Neural Network
BCC	Basal Cell Carcinoma
BKL	Benign Keratosis
BSD	Berkeley Software Distribution
CDSS	Clinical Decision Support System
CNN	Convolutional Neural Network
DF	Dermatofibroma
DL	Deep Learning
DoD	Definition of Done
EMA	European Medicines Agency
GUI	Graphical User Interface
HPND	Historical Permission Notice and Disclaimer
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
IMDRF	International Medical Device Regulators Forum
IRMC	Iterative Risk Management Cycle
ISO	International Organization for Standardization
IVD	In Vitro Diagnostic Device
IVDR	In Vitro Diagnostic Medical Device Regulation
LGPL	Lesser General Public License
MDCG	Medical Device Coordination Groups
MDR	Medical Device Regulation
MDSW	Medical Device Software
MEL	Melanoma
ML	Machine Learning
NV	Melanocytic Nevus
PSFL	Python Software Foundation License
QMS	Quality Management System
RM	Risk Management
SaMD	Software as a Medical Device
SLC	Software Life Cycle
SOP	Standard Operating Procedure
UDI	Unique Device Identification
UE	Usability Engineering
UEIDC	Usability Engineering Iterative Design Cycle
UI	User Interface
UOUP	User Interface of Unknown Provenance
VASC	Vascular Lesion

List of Figures

2.1. Sample Dermatoscopic Skin Lesion Images from the HAM10000 Dataset. . .	3
2.2. Image of the Used Digital Dermatoscope with a Built-In Polarization Filter. .	5
2.3. Outline of a Typical CNN Architecture.	6
3.1. Quality Management System Components.	22
3.2. Software Life Cycle Components.	23
3.3. Risk Management System Components.	26
3.4. Usability Engineering Components.	28
4.1. Process Description Plan as a Linear Sequential Model.	36
4.2. Software Development Plan.	37
4.3. Extract of the Risk Policy: The Severity Classes.	38
4.4. Extract of the Risk Policy: The Resulting Risk Matrix.	39
4.5. High-Level Functional Requirements from the System and Software Require- ments Specification.	40
4.6. Extract from the Usability Test Protocol Table of the Summative Evaluation. .	42
4.7. Iterative Risk Management Cycle 1: Hazard Analysis.	43
4.8. Iterative Risk Management Cycle 1: Risk Analysis.	44
4.9. Iterative Risk Management Cycle 2: Risk Evaluation.	44
4.10. Iterative Risk Management Cycle 2: Risk Matrix.	45
4.11. Iterative Risk Management Cycle 3: Not Acceptable Risks and the Descrip- tion of Their Control Measure.	45
4.12. Iterative Risk Management Cycle 3: Acceptable Risks and the Description of Their Control Measure.	45
4.13. Risk Monitoring: Final Updated Risk Table.	46
4.14. Risk Monitoring: Updated Risk Matrix.	46
4.15. Visualization of the Guideline Applications in Relation to the Different Stages. .	48
4.16. Screenshot of the Welcome Page of the Software for Skin Cancer Detection. .	54
4.17. Screenshot of the Data Input Page of the Software for Skin Cancer Detection. .	54
4.18. Screenshot of the User History Page of the Software for Skin Cancer Detection. .	55
4.19. Screenshot of the Image Capturing Page of the Software for Skin Cancer Detection.	55

List of Tables

2.1. Exemplary Requirements and Their Descriptions Under the MDR from the Johner Institut.	10
2.2. IVDR Key Aspects and Their Description.	11
2.3. ISO 9001 Topics and Their Description.	12
2.4. ISO 62304 Topics and Their Description.	12
2.5. ISO 82304 Topics and Their Description.	13
2.6. ISO 25010:2011 Topics and Their Description.	13
2.7. ISO 14971 Topics and Their Description.	14
2.8. IEC 62366 Topics and Their Description.	15
3.1. Python Libraries Used for the Skin Cancer Detection Software Implementation.	17
3.2. Academia-Tailored Guidelines and Their Activities	21
4.1. Database: Table <i>users</i> Attributes.	49
4.2. Database: Table <i>imagesAndResults</i> Attributes.	50

1. Introduction

In the ongoing struggle against cancer, skin cancer emerges as one of the most common and growing concerns for global public health, with an estimated 1.5 million new cases diagnosed worldwide in 2022, according to the World Health Organization [1]. It presents a unique challenge: while highly treatable when identified early, it can lead to severe consequences if diagnosed too late [2]. This discrepancy underscores the urgent need for enhanced methods in early detection and diagnosis, which has become an important point of research and innovation in the medical field.

The statistics are evident. As stated by experts like the US Dermatology Partners, one of the largest dermatology practices in the US [2], or Tsao et al. [3], early skin cancer detection and treatment can lead to a promising five-year survival rate of about 99 % for patients with a melanoma, the most dangerous and malignant form of skin cancer, if in situ (in initial stage). This rate drops dramatically to less than 15 % when the cancer starts spreading during the later stages 3 or 4, turning into a metastatic melanoma. This disease progress highlights the critical role of an early diagnosis in improving patient outcomes.

In recent years, the medical community has increasingly turned to artificial intelligence (*AI*) and machine learning (*ML*) to revolutionize the detection and treatment of skin cancer [4], [5]. These technologies aim to improve skin cancer diagnostics by enhancing accuracy and efficiency. For example, a study presented at the annual European Academy of Dermatology and Venereology (*EADV*) Congress 2023 [6] has revealed a significant improvement in skin cancer detection with new AI software, correctly detecting a remarkable 99.5 % of all skin cancers. Notably, for melanomas, the detection rate in that study reached a flawlessly accurate 100 %.

Machine learning algorithms in particular are already common practice in medicine, providing an objective and accurate framework to support healthcare professionals in many areas [7]. Yet, the integration of such technologies into clinical settings faces significant hurdles, including regulatory challenges and the need for comprehensive validation to ensure their reliability and effectiveness. Generally speaking, developing a software for medical use is a process linked with great effort, requiring a lot of documentation, attention to detail, and substantial time before it can be transferred to the medical industry [8]. Stringent regulations, such as the Medical Device Regulation (*MDR*) or the In Vitro Diagnostic Medical Device Regulation (*IVDR*), exist for Medical Device Software (*MDSW*) in addition to internationally recognized International Organization for Standardization (*ISO*) standards on how to implement these regulation requirements. It is difficult to constantly keep track of all the requirements and constraints, especially for academic institutions, which often operate with limited resources [9].

This project faces the challenge of creating a regulatory-compliant, clinically useful medical software. It follows a structured approach to developing a clinical decision support system (*CDSS*) for skin cancer detection, resembling the research project "the virtual doctor", an interactive AI-based CDSS [10], and leveraging a machine learning model utilizing the HAM10000 dataset, a comprehensive collection of dermoscopic images [11], [12]. By applying four academia-tailored guidelines encompassing quality management systems, software life cycle, risk management, and usability engineering, this project not only demonstrates the feasibility of a guideline-based approach to medical software development, but also illustrates the potential of machine learning to enhance diagnostic processes with the herein developed CDSS for physicians, which can calculate risk scores for seven of the most prominent skin lesion types from captured photographs in a matter of seconds. The relevance of this work extends beyond skin cancer detection. It highlights the importance of structured development processes in overcoming barriers to regulatory compliance of medical software and contributes to future advancements in medical diagnostics employing the multifaceted capabilities of AI and other technologies.

In the subsequent chapters, the thesis provides a more detailed background on skin cancer, the diagnostic process, and machine learning. It then introduces relevant international medical software standards and the four streamlined guidelines, before delving into the materials and methodology employed in implementing these guidelines and developing the CDSS. In the end, the results of the implementations are presented and discussed.

2. Background

2.1. Skin Lesions and Diagnosis

Skin cancer is one of the most common forms of cancer that occurs worldwide [13]. It is estimated that one in five Americans will develop skin cancer in their lifetime, and approximately 9,500 people in the U.S. receive a skin cancer diagnosis every day [13]. In Germany, according to a report published by the German Federal Statistical Office (*Destatis*), a significant increase in skin cancer cases and related hospital treatments is observed [14]. In 2021 alone, the number of people hospitalized with a skin cancer diagnosis reached 105,700, marking a 75 % increase from 2001. Additionally, skin cancer deaths have increased by 55 % since 2001, while the number of deaths due to cancer diseases overall increased by only 10 %, highlighting a significant rise in mortality specifically related to skin cancer compared to other forms of cancer [14].

This thesis focuses on the seven most relevant skin lesion classifications [15]. These are:

- actinic keratosis (*akiec*),
- basal cell carcinoma (*bcc*),
- benign keratosis (*bkl*),
- dermatofibroma (*df*),
- melanoma (*mel*),
- melanocytic nevus (*nv*), and
- vascular lesion (*vasc*).

Although other lesions exist, more than 95 % of all cases in clinical practice are covered by those mentioned above [12]. Figure 2.1 represents some image samples for each of those classes.



Figure 2.1.: Sample Dermatoscopic Skin Lesion Images. Taken from the HAM10000 dataset [12], in order from left to right: *akiec*, *bcc*, *bkl*, *df*, *mel*, *nv*, *vasc*.

Actinic keratoses (*akiec*) are the most common precursors of skin cancer, caused by prolonged sun exposure. On the skin, they usually appear as a rough, scaly patch, that if

left untreated, often evolves into a malignant form of skin cancer, e.g. basal cell carcinoma (*bcc*) [16]. Basal cell carcinomas belong to the non-melanoma skin cancers, usually appearing as a slightly transparent bump. While slow-growing and rarely metastasizing (spreading to other parts of the body), it can be locally aggressive and destroy surrounding tissue, including even bones [17]. Therefore, it must be treated early to avoid extensive tissue damage and scarring [18].

The exact cause of benign keratoses (*bkl*), also called seborrheic keratoses, remains unknown, although genetic mutations are suspected, as they have been identified in some cases [19]. This lesion type varies in color and size, often appearing as waxy and "glued-on", and whilst some patients complain of cosmetic unappealingness and irritation, a removal is not necessary since they are harmless [19]. Dermatofibromas (*df*) are benign skin lesions that present as a firm growth beneath the skin's surface following a trauma or an insect bite [20]. Since those are asymptomatic and harmless, in most cases no therapy is required [21].

The most dangerous, malignant form of skin cancer is a melanoma (*mel*) because of its high metastasizing potential [22]. It originates from melanocytes, which are cells that produce melanin, a skin pigment [23]. Those cells are also the source of melanocytic nevi (*nv*), the scientific name for skin moles [24]. Melanomas, while less common than *bccs*, are likely to become life-threatening if not discovered and treated in early stages. Early detection can prevent a deadly cause and increase chances of survival [2], [25].

The generic term for anomalies present at birth or in early childhood is vascular lesion (*vasc*) [26]. Lesions of that type vary in size, shape, and color, and range from simple benign birthmarks to other malignant malformations, with therapeutic interventions depending on the symptoms [26].

It is not rare that a skin lesion is classified incorrectly, too late, or not at all, possibly having a deadly consequence [27]. Therefore, it is important to correctly classify an existing skin lesion in its early stages.

Several methods exist in diagnosing skin cancer and clinically examining skin lesions, including incisional skin biopsies, confocal microscopies, computer-aided diagnoses, mole mappings, and most importantly, dermatoscopies [28]. The last and commonly used method involves the utilization of a dermatoscope (or dermoscope) [27], which is a handheld device that provides a magnified view of the skin, thus allowing for a clear inspection of skin lesions [29]. Some devices also come with a built-in polarization filter that nullifies surface light reflections, resulting in an even more enhanced view of the lesion and minimizing the occurrence of misdiagnoses. The dermatoscope used in this thesis is depicted in Figure 2.2.

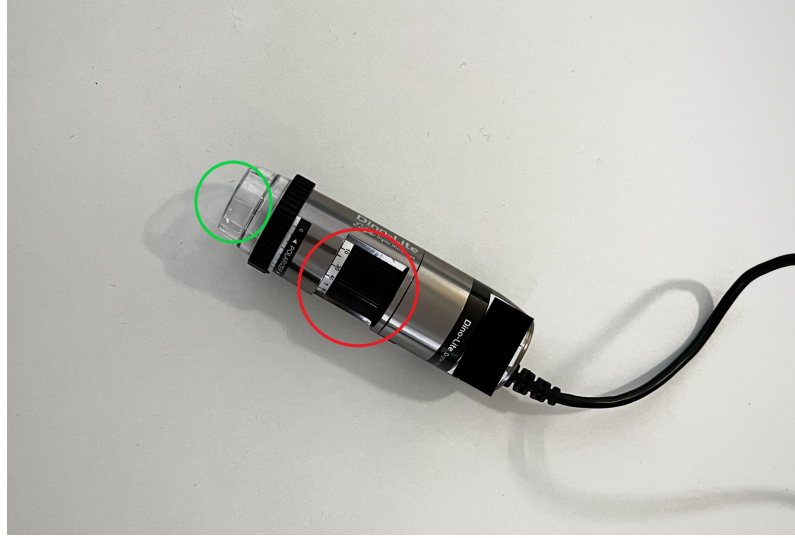


Figure 2.2.: Image of the Used Digital Dermatoscope with a Built-In Polarization Filter That Can Capture an Image. Polarization angle can be adjusted in the red circled area. The green circle shows the location of the built-in camera and should be placed on the to-be-tested skin tissue.

The first step in the identification of skin lesions is a visual examination, which includes the practice of dermatoscopy. If a suspicious lesion is identified, raising concerns about skin cancer, the physician may recommend a skin biopsy. During the biopsy, a small sample of the tissue is extracted and microscopically examined more thoroughly in order to either confirm or rule out the presence of skin cancer [30].

2.2. Machine Learning

Machine learning (*ML*) is a subset of artificial intelligence (*AI*) that focuses on systems that learn from data. It involves training models on given datasets to make predictions or decisions without being explicitly programmed for specific tasks [31].

After collecting and preprocessing data, which includes, amongst others, cleaning data by removing duplicates, correcting errors, and handling missing data, a model is chosen and trained [31]. The general training process involves optimizing model parameters to minimize a cost function, which measures the difference between the model's predictions and the actual outcomes [32].

Following the training of the model, its performance is evaluated, and based on the results, the model is optimized [31].

2.2.1. Deep Learning and Convolutional Neural Networks

Deep learning (*DL*) aims to mimic the functionality of the human brain to analyze big data [33], [31]. The machine learning pipeline that is used for the skin cancer detection software operates on an Artificial Neural Network (*ANN*), more specifically a Convolutional Neural Network (*CNN*), which is a deep learning algorithm that specializes in solving image classification or object recognition tasks [34], [35].

CNNs' versatility and robustness in handling image data have led to their widespread adoption in various fields, including medical image analysis, where accurate and efficient image classification is crucial [7], [36].

The architecture of a CNN is generally composed of at least four layers, namely an input layer that receives the image, multiple hidden layers that process the image, and an output layer that outputs the classification [36], as visualized in Figure 2.3. The hidden layers typically include multiple convolutional layers and pooling layers, and in the end fully connected layers.

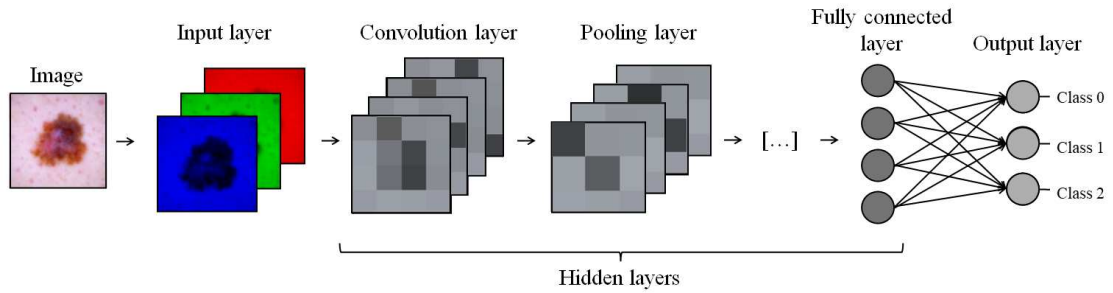


Figure 2.3.: Outline of a Typical CNN Architecture. Exemplary, a melanoma image from the HAM10000 dataset [12] is the input for the input layer. Colored images are rendered in three RGB channels. Then, convolutional layers use filters on the input to capture spatial patterns such as edges, textures, and other image features, while pooling layers reduce the spatial size of the representation, improving computational efficiency and reducing overfitting [36]. The bracketed dots illustrate that multiple sequences of convolution and pooling layers may be integrated. In the end, fully connected layers use these extracted features to classify images into different classifications in the output layer [36].

At the forefront of image processing within a CNN are the convolutional layers, where the detection of features like edges or textures occurs, as shown in Figure 2.3 [36]. These layers utilize filters or kernels, small matrices, to traverse the image to highlight specific features, such as edges or textures, by emphasizing certain patterns and de-emphasizing others, depending on the weights that determine the importance of features in these filters [36], [37]. Throughout the training process, the network optimizes these filters to better capture the most relevant features [36], [34]. The stride, the distance the filter moves across the image, is critical in determining the output feature map's size [36]. A smaller stride yields a more

detailed map, while a larger stride reduces the map's size and processing time, potentially costing some detail loss [36].

As these filters slide across the image, they perform a convolution operation. This operation is a mathematical process that combines the pixels of the image with the values of the filter to generate a feature map, represented by the 2D-equation (2.1) or the 3D-equation (2.2) [38].

$$y[i, j] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h[m, n] \cdot x[i - m, j - n] \quad (2.1)$$

In this equation, x is the input image matrix to be convolved with h , which represents the kernel (filter) matrix. Together they result in a new matrix y , which equals the output image. (i, j) are the coordinates on the image output feature map, while those of (m, n) deal with the kernel's [38], [39].

For a 3D convolution, an additional dimension exists, so the equation incorporates a sum over the third dimension k [40]:

$$y[i, j, k] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \sum_{o=-\infty}^{\infty} h[m, n, o] \cdot x[i - m, j - n, k - o] \quad (2.2)$$

In this expanded equation y again is the output matrix at position (i, j, k) , while x represents the input image at a position offset by (m, n, o) from (i, j, k) , and h is the matrix of the 3D kernel at position (m, n, o) . The summations over m , n , and o are for the convolution over the three dimensions, integrating through the entire extent of the kernel in each dimension [40].

Padding allows every element of the input, including the edges, to be the center of the kernel operation at some point, ensuring full coverage of the input image. It is an essential technique in CNNs, involving the addition of extra pixels around the edges of the image to ensure filters fully cover the image edges, maintaining the spatial dimensions of the output feature map and preserving the integrity of the image's features [36].

The convolutional layers are followed by activation functions, which introduce non-linearity to the network, enabling it to capture complex patterns. The Rectified Linear Unit $\text{ReLU}(x)$ is a common choice, maintaining positive values while converting negative values to zero, thus filtering out less significant features [41]. Its function is defined in equation (2.3). Other common activation functions are the sigmoid function $\sigma(x)$ (2.4) or the hyperbolic tangent $\tanh(x)$ (2.5).

$$\text{ReLU}(x) = \max(0, x) \quad (2.3)$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.4)$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.5)$$

Pooling layers are essential for simplifying information and reducing feature map dimensionality [36]. By downsampling and effectively reducing overfitting, pooling layers ensure that the model does not merely perform well on familiar data but also possesses the capacity to transfer its knowledge to new data. These layers allow the model to extract and learn essential patterns, moving from sole memorization towards a deeper understanding. This process is crucial for the model’s ability to apply its knowledge to unseen data, addressing the issue of non-generalization and enhancing overall performance [31]. Max pooling, for instance, retains the maximum value within each patch of the feature map, highlighting the most significant features [42].

As the image progresses through the network, it eventually reaches fully connected layers, which integrate all the features learned to classify the image into categories [36]. Prior to this stage, normalization techniques might be applied to facilitate smooth and efficient training [31].

Through the convolutions, activation, pooling, and fully connected layers, CNNs are able to automate the process of feature extraction and classification. They can learn to identify and highlight the most informative features of images, becoming a foundational technology in modern image processing and computer vision. Unlike other machine learning algorithms that require manual feature extraction, CNNs can directly learn relevant features from raw input data, e.g. from medical images [43], making them exceptionally suited for complex tasks like skin cancer detection.

2.3. Medical Software

Medical software encompasses a broad range of applications designed to support healthcare professionals and patients in various aspects of healthcare, including diagnosis, treatment, monitoring, and administration [44]. This broad category contains two relevant key concepts, namely Medical Device Software (*MDSW*), which is only used in the EU, and Software as a Medical Device (*SaMD*), which is not used in the EU [45], [46], among others.

Both terms are often used interchangeably in medical environments, even when they are not the same [46], [45]. Understanding the used terms is crucial in grasping the evolving area of healthcare technology.

2.3.1. Medical Device Software

Software that is an integral part of a medical device is typically referred to as medical device software [45]. This software is designed to support the device’s intended medical purpose, whether it’s diagnostic, therapeutic, or monitoring [46]. It can be embedded in hardware devices [47], e.g. pacemakers or MRI machines, where the software plays a critical role in the functioning of the device [46]. The primary characteristic of MDSW is that it cannot typically operate independently without the medical device it supports, e.g. X-ray software [46]. Regulatory bodies, like the European Medicines Agency (*EMA*) in the European Union, strictly regulate MDSW due to its direct impact on patient health and safety [44]. Compliance with these regulations ensures that MDSW meets stringent safety, quality, and efficacy standards.

2.3.2. Software as a Medical Device

Unlike MDSW, SaMD is software that functions as a medical device in its own right and is intended to be used ”for one or more medical purposes that perform these purposes without being part of a hardware medical device”, as defined by the International Medical Device Regulators Forum (*IMDRF*) [47]. SaMD can run on general-purpose devices (e.g., smartphones, tablets, or PCs) and doesn’t necessarily require dedicated hardware to operate [46]. Examples include software that helps diagnose skin lesions from photographs, or consumer apps like symptom checkers developed by Ada, a global health company that offers an AI-powered health platform designed to help users understand their health symptoms and navigate to the appropriate care [48].

2.4. International Standards for Medical Software

All medical devices and software must undergo strict regulations and comply with international standards in different sectors. The most important and relevant internationally recognized norms and their requirements are explained in the following sections.

2.4.1. The Medical Device Regulation

The Medical Device Regulation (*MDR*) must be taken into account by manufacturers of medical products, but also by hospitals and merchants, if they want to distribute their

products in the European Union [49]. Table 2.1 lists some example requirements by the MDR.

Table 2.1.: Exemplary Requirements and Their Descriptions Under the MDR from the Johner Institut [49].

Requirement	Description
Quality Management System (<i>QMS</i>)	Manufacturers require a QMS for the development, production, and monitoring of products on the market.
Risk Management (<i>RM</i>) System	The RM must ensure that the benefit of the product's life cycle is acceptable in regards to possible risks.
Classification of Devices	Manufacturers must determine the risk class of every product. (Class I to Class III, with I the lowest and III the highest risk)
Conformity Assessment	Manufacturers must ensure that their devices meet the requirements set out in the regulation.
Clinical Evidence	Manufacturers must provide clinical data demonstrating the safety and performance of their devices.
Unique Device Identification (<i>UDI</i>)	Devices must be identifiable to facilitate traceability throughout the supply chain.
Post Market Surveillance and Vigilance	Manufacturers must monitor their medical products' performance and safety once they are on the market, and report serious incidents.
Responsible Person for Regulatory Compliance	Manufacturers must assign a person responsible for regulatory conformity.

In order to fulfill the MDR requirements, the academia-guidelines orient themselves on several International Organization for Standardization (*ISO*) certifications.

2.4.2. The In Vitro Diagnostic Regulation

One of many regulatory frameworks by the EU, the In Vitro Diagnostic Regulation (*IVDR*), aims to ensure the safety, performance, and quality of In Vitro Diagnostic Devices (*IVDs*) and enhance the transparency and traceability of the development and the supply chain [50]. IVDs are used to test tissue or blood samples outside the body for the diagnosis, prevention, monitoring, or treatment of medical conditions, infections, and diseases [51]. The skin cancer detection software is an example for an IVD. Important aspects are listed and described in Table 2.2.

Table 2.2.: IVDR Key Aspects and Their Description [52].

Chapter	Description
Scope and Definitions	Scope of IVDs and definitions for various terms used in the regulation.
Most Important Requirements	Includes provisions for design, risk management, documentations, etc. to ensure the safety and performance of the IVD.
Traceability of Products	UDI system to facilitate traceability and improve the identification and tracking of IVDs.
Notified Bodies	Requirements, tasks, and obligations for notified bodies responsible for assessing conformity with the regulation.
Conformity Assessment	Procedures and requirements for demonstrating regulation conformity.
Performance Evaluation	Process for evaluating IVDs' performance, design, and criteria.
Post-Market Surveillance	Establishment and maintenance of post-market surveillance systems to monitor the performance and safety of released devices.
Cooperation between Member States, Medical Device Coordination Groups (<i>MDCG</i>) and Other Experts	Collaboration, harmonization, and effective implementation of the IVDR across the EU regulatory network for transparency.
Confidentiality, Data Protection, Penalties	Provisions for protecting confidential information, compliance with data protection regulations, and penalties for non-compliance with the IVDR requirements.

2.4.3. Quality Management System

In order to document responsibilities, processes, regulations, and procedures, and to ensure compliance and consistency throughout the development life cycle, a standardized system, namely a quality management system (*QMS*), is employed. To develop a QMS, the ISO 9001 is applied. It is the most widely spread generic standard for quality management and focuses on the establishment, maintenance, and most importantly continuous improvement of a QMS by covering all aspects that manage the quality of an organization's products and services [53], including the aspects in Table 2.3.

Almost identically, the ISO 13485 incorporates mandatory QMS requirements, with the difference being the specific focus on medical device industries' efficiency and safety throughout the devices complete life cycle [54].

Table 2.3.: ISO 9001 Topics and Their Description [53].

Topic	Description
Context of the Organization	External and internal factors that affect the organization's ability to achieve intended QMS results.
Leadership	The importance of leadership for QMS.
Planning	Measures defined to achieve an organization's quality objectives and effectiveness improvement.
Support	Resources, competence, awareness, communication and documented information issues.
Operation	Planning, implementation and controlling of processes to meet customer requirements and improve satisfaction.
Performance Evaluation	The monitoring, measurement, analysis, and evaluation of the QMS's performance and effectiveness.
Improvement	Increasing the effectiveness of QMS continuously.

2.4.4. Software Life Cycle

A cost-effective and time-efficient process used to design and build high-quality software is called a software life cycle (*SLC*). It emphasizes the importance of each phase from planning to deployment. Both the IEC 62304 and IEC 82304 standards address software development used in the healthcare industry [55], [56]. While the former focuses on life cycle requirements specifically for MDSW, ensuring it is developed and maintained safely [55], the latter broadens the scope to include general health software products, detailing safety and security requirements across the product's life cycle for manufacturers [56]. The following Table 2.4 and Table 2.5 list some of the relevant aspects covered by the standards.

Table 2.4.: ISO 62304 Topics and Their Description [57].

Topic	Description
Software Development Process	Defines steps for the development of medical software, ensuring safety and reliability.
Software Maintenance Process	Guide to maintaining software post-deployment, including updating and safety monitoring.
Software Risk Management Process	Integration of RM.
Software Configuration Management Process	Details on how to handle configuration of software.
Software Problem Resolution Process	Definition of an approach to solving problems identified during either development or post-market.

Table 2.5.: ISO 82304 Topics and Their Description [58].

Topic	Description
Software Product Requirements	Defines general requirements for health software products.
Software Life Cycle Processes	Details life cycle requirements for the development, maintenance, and decommissioning of health software.
Product Validation	Describes validation processes involved in realizing health software products from conception to release.
Product Identification	Outlines methods for uniquely identifying health software throughout their life cycle (similar to UDI).
Post-market Activities	Focus on monitoring and improving the product's safety after release.

Another standard is the ISO 25010:2011, which has recently been withdrawn. It defines a set of quality characteristics for software product and service quality, similar to the newer ISO 25010:2023 version [59] which has been released in 2023. The ISO 25010:2011 version was mentioned in one of the later introduced academia-tailored guidelines and is therefore especially relevant, including aspects as listed in Table 2.6.

Table 2.6.: ISO 25010:2011 Topics and Their Description [60], [61].

Topic	Description
Functional Suitability	Requires completeness, correctness, and appropriateness of software functions.
Performance Efficiency	Evaluates the efficiency to the performance in relation to used resources and conditions.
Compatibility	Evaluates software co-existence with other software and interoperability efforts.
Usability	Requires the software to be easy to use, understandable and attractive to the user.
Reliability	Defines requirements to evaluate the ability of the software to maintain its performance under stated conditions for a specific period of time.
Security	Describes the ability of the software to protect data regarding integrity, confidentiality, and authenticity.
Maintainability	The software should be easy to modify, improve, correct or adapt to changes in requirements and/or in the environment.
Portability	Requires the software to be easy to install and to exchange depending on the environment of use.

2.4.5. Risk Management

The supplementary standard ISO 24971 assists manufacturers in applying and interpreting the ISO 14971, an international fundamental standard for the application of risk management (*RM*) to medical products throughout their life cycle. RM involves the systematic estimation and evaluation of risks together with the identification of strategies in order to avoid or minimize their occurrence probability and impact, thereby safeguarding the software development process [62], [63]. ISO 14971 activities and their description can be found in Table 2.7.

Table 2.7.: ISO 14971 Topics and Their Description [64].

Topic	Description
Risk Management Plan	Planning of foreseeable RM activities with risk identification, role and responsibility assignment, and risk acceptance.
Risk Analysis	Identification of safety characteristics of a medical device and definition of hazardous situations and risk severity & occurrence.
Risk Evaluation	Evaluation of identified risks and placement into acceptable or unacceptable risks.
Risk Control	Definition of appropriate control measures for identified risks.
Evaluation of Overall Residual Risks	A risk evaluation of residual risks and placement into acceptable and unacceptable risks.
Risk Management Review	Confirmation that all overall residual risks are acceptable and all steps in RM plan are completed.
Production and Post Production Information	Establishment of a collection of production-related risks events and documentation during and post-production.

2.4.6. Usability Engineering

Lastly, usability engineering (*UE*) is a process that concentrates on developing the usability of a software, and, similar to RM, minimizing error occurrences. The IEC 62366 standard family consists of the IEC 62366-1 and IEC 62366-2. The former applies UE to medical devices to enhance their usability and safety for users, while the latter provides systematic guidance on it [65], [66]. The IEC 62366 bundle ensures that medical devices are designed and developed in a way that minimizes the risk of user error, enhances user satisfaction, and ultimately improves patient safety and outcomes. Main topics are listed in Table 2.8.

Table 2.8.: IEC 62366 Topics and Their Description [65], [67].

Topic	Description
General Requirements	Sets expectations for integrating UE into the medical device life cycle.
Usability Engineering Process	Describes a detailed process for applying UE throughout the design and development of medical devices.
User Interface of Unknown Provenance (<i>UOUP</i>)	User interfaces (<i>UIs</i>) brought into the design without full usability engineering processes applied.
Usability Specification	Documenting usability goals and safety requirements that the medical device needs to meet.
Usability Validation	Details methods and criteria for validating that the device meets the specified usability and safety requirements.
Risk Management	Integrates UE in risk management for the reduction of use-related risks.
User Interface Design	Focuses on designing intuitive user interfaces that reduce the risk of user error and are effective.

2.5. Academia-tailored Guidelines for Medical Software

While modern healthcare essentially depends on software, the strict regulations on medical devices result in barriers to technology transfer from research institutes to the medical industry [9]. In recent years, multiple guidelines have been developed to remedy this cavity by covering different aspects of medical software development.

Four guidelines were selected for this thesis, which were developed by Prof. Dr. Dominik Heider and his coworkers, covering the following:

- Quality Management System guideline, which is based on the ISO 9001 and ISO 13485,
- Software Life Cycle guideline, based on the ISO 25010:2011 and both the IEC 62304 and IEC 82304,
- Risk Management guideline, which implements the ISO 24971 and ISO 14971,
- Usability Engineering guideline, in conformity with the IEC 62366 standard family.

These guidelines discuss various connected aspects, as elucidated later in the Materials section. They are independent and yet intertwined with each other, for example activities of the UE guideline refer to activities in the RM guideline, and aspects defined in the QMS guideline apply to all guidelines.

3. Materials and Methodology

All materials used in this project to successfully develop the software with the guidelines are listed below and expounded upon in the following sections.

- The four provided guidelines:
 - Guideline *"Fostering reproducibility, reusability, and technology transfer in health informatics"*, authored by Anne-Christin Hauschild, Lisa Eick, Joachim Wienbeck, and Dominik Heider and published on July 1st 2021 in iScience [68].
 - Guideline *"Guideline for software life cycle in health informatics"*, authored by Anne-Christin Hauschild, Roman Martin, Sabrina Celine Holst, Joachim Wienbeck, and Dominik Heider and published on November 9th 2022 in iScience [69].
 - Unpublished manuscript *"Guideline for Risk Management in Software Health and Medical Applications"*, last accessed in April 2024 and authored by Roman Martin, Anne-Christin Hauschild, Robin Gottschalk, Sandra Clemens, Joachim Wienbeck, and Dominik Heider [70].
 - Manuscript *"Manuscript on Usability Process"*, authored by Dominik Heider, Anne-Christin Hauschild, Joachim Wienbeck, Roman Martin, Sandra Clemens, and Vanessa Klemt and published on July 31st 2023 as an EU deliverable research report of the FeatureCloud project [71]. Based on Vanessa Klemt's master's thesis *"Usability Engineering Guideline for Software as a Medical Device - Implementing an Interactive xAI Platform supporting Medical Decision-Making"*, submitted on September 14th 2021 at the Philipps University of Marburg [72].
- Integrated Development Environment (*IDE*): JetBrains PyCharm, version 2023.3.2.
- Documentation platform: Confluence and Jira.
- Code documentation platform: HHU GitLab.
- Python packages as described in Table 3.1.
- Qt Designer software, version 5.11.1.
- Machine learning pipeline for skin cancer detection from Dmitry Degtyar's bachelor's thesis *"Machine Learning based skin cancer screening"*, submitted on July 8th 2023 at the Philipps University of Marburg [11]
- Personal Acer computer with six CPU cores and 16 GB main memory.
- A dermatoscope, model *"Dino-Lite DermaScope Polarizer HR"* was used, funded by the Hessian Center for Artificial Intelligence.

3.1. Software Specification

3.1.1. Python Libraries and Packages

Different python packages were used for the implementation of the skin cancer detection software. Table 3.1 lists all packages used including their licenses, version, and purpose. All licenses mentioned are permitted for use in academic settings.

Table 3.1.: Python Libraries Used for the Skin Cancer Detection Software Implementation.

Package	Version	License	Purpose
torch	1.12.1	Modified BSD	Used for creating and training the ML-pipeline.
torchvision	0.15.2	Modified BSD	Used to access pretrained CNN models.
PIL (Pillow)	9.4.0	HPND	Used for opening and saving images.
sqlite3	3.41.2	Public Domain	Used for the databases. Does not require a separate server process.
sys	3.10.9	PSFL	Used for accessing command-line arguments, managing the Python path, or controlling script termination.
unittest	3.10.9	PSFL	Used for constructing and running tests.
os	3.10.9	PSFL	Used for managing file paths, directories, and saving and deleting files.
PyQt5	5.15.7	LGPL	Used to develop the graphical user interface (<i>GUI</i>) for the application (creating windows, dialogs, and widgets for user interaction).
cv2 (opencv-python)	4.8.1	3-clause BSD	Used for image processing tasks, such as reading and displaying images, or using the camera.
datetime	3.10.9	PSFL	Used for manipulating dates and times.
re	2.2.1	PSFL	Used for regular expression matching operations.
classification-pipeline	1.0	Public Domain	ML pipeline.

3.1.2. Qt Designer

Qt Designer is a tool that comes with the Qt framework, used for Drag-and-Drop-designing and building GUIs with Qt Widgets. It allows developers to create forms in a what-you-see-is-what-you-get (*WYSIWYG*) manner, making it easier to arrange graphical components like buttons, labels, or sliders on the application's windows or dialogs [73].

The built-in property editor allows developers to modify the properties of selected widgets. Properties can include things like the widget's size, font, and background color, among others. Furthermore, signals and slots can be used in the Qt framework for the communication between objects. The signal and slot editor in Qt Designer makes it easy to connect widgets and define their interactions without manual coding. Developers can also preview their designs in different styles and on different platforms directly from Qt Designer [73].

The created GUI layout is saved as a .ui file, which is an XML file describing the properties and layout of the widgets in the interface. This .ui file can be loaded into an application at runtime or converted into Python code using a tool like pyuic for PyQt applications [73].

The software simplifies the process of GUI design, making it more visual and less code-intensive. It was primarily used for the software development and usability engineering process in this thesis.

3.1.3. Documentation Platforms

Documenting code and guideline activities was done with the help of GitLab, Confluence, and Jira as respective documentation platforms.

GitLab is a DevOps platform that combines source code management (*SCM*) and continuous integration/continuous deployment (*CI/CD*) tools to streamline software development and collaboration. Source code was stored and managed with Git which was provided by the Heinrich-Heine University Düsseldorf.

Confluence on the other hand is a content collaboration tool that helps teams create, share, and organize documentation and project plans. In this project, Confluence is mainly used for documenting the guideline procedures, as it allows the creation of so called spaces (wikis). It can be connected and integrated with Jira, which is a project management tool designed for issue and project tracking, enabling teams to organize tasks, bugs, and feature requests in an efficient manner.

Together, these tools support comprehensive project planning, development, and collaboration efforts.

3.2. Machine Learning Pipeline

The core objective of the ML model used for the skin lesion classification in this thesis is to develop a highly accurate classification system for different types of skin lesions, utilizing the HAM10000 dataset as the primary material for training and validation of the model [11]. The cited thesis evaluates several deep learning architectures, including variations of DenseNet and ResNet, among others. Key aspects of the thesis are summarized in the following sections.

3.2.1. Skin Lesion Dataset: HAM10000

The HAM10000 (short for *Human Against Machine with 10000 Dermoscopic Images*) dataset is a collection of dermoscopic images that are instrumental for the classification of skin lesions [12]. It has over 10,000 images spanning seven distinct generic categories of skin lesions, including those mentioned in the background: actinic keratoses, basal cell carcinomas, benign keratoses, dermatofibromas, melanomas, melanocytic nevi, and vascular lesions. Each image within the dataset has undergone rigorous clinical verification, ensuring a high degree of accuracy and reliability for training machine learning models, and was collected from different clinical and research institutions [12], [11].

3.2.2. Preprocessing

In the cited thesis [11], the preprocessing of dermoscopic images for the development of a skin lesion classification pipeline involved several steps to enhance the quality and uniformity of the input data. First, the raw images from the HAM10000 dataset were transformed into structured data frames. These frames contained representations of images as multidimensional vectors of numerical pixel values, paired with corresponding disease class identifiers. The dataset was then divided into training and validation subsets while maintaining a uniform distribution across the classes [11].

An oversampling technique called random oversampling [74] was applied to all classes except the majority class to equalize the number of images across classes. Afterwards, the images underwent a series of augmentation processes, including cropping, rotation, scaling, and distortion, to generate a more diverse set of training data. Additionally, normalization procedures based on the equation (3.1) were applied to each image to scale the pixel values, ensuring consistency across the dataset [11].

$$\text{output[channel]} = \frac{\text{input[channel]} - \text{mean[channel]}}{\text{std[channel]}} \quad (3.1)$$

In the equation, output[channel] represents the resulting normalized channel value (red, green or blue channel), while input[channel] stands for the raw channel values. The mean[channel] is the mean value, and std[channel] is the standard deviation value of each corresponding channel, calculated for the whole dataset [11], [75], [76], [77].

3.2.3. Transfer Learning

The model utilizes transfer learning, which is a machine learning technique where a model developed for a particular task is reused as the starting point for a model on a second

task, thereby reducing dependence on large amounts of training data and accelerating the learning process [78], [79]. It was employed to leverage knowledge from the large, diverse ImageNet dataset to improve performance despite the relatively smaller size of the HAM10000 dataset, thereby enhancing the models' ability to generalize from the training data to new, unseen images. This approach is important in contexts where collecting large amounts of labeled data is challenging, for example in medical imaging [11].

3.2.4. Training Process

The training process for the machine learning model involves several detailed methodological approaches, including data preparation, fine-tuning of pre-trained models, and the application of transfer learning using CNNs.

The models that were trained included several advanced CNN architectures like DenseNet, ResNet, InceptionV3, AlexNet, and VGG, with variations of DenseNet and ResNet explored for their efficacy in classification tasks [80], [11]. The training utilized a cross-entropy loss function to calculate the difference between the predicted probability distributions and the actual label distributions. Stochastic Gradient Descent (*SGD*) was used as an optimization strategy, adjusting the model's parameters in small steps to minimize the loss function. It uses sigmoidal annealing to adjust the learning rate, with the equation depicted in (3.2) [81], [11]. Specifically, the learning rate is gradually increased during an initial warm-up phase, which constitutes 10 % of the total training epochs, starting from a lower limit (η_{low}) of 0.001 to an upper limit (η_{up}) of 0.1.

$$\eta_t = \eta_{\text{low}} + (\eta_{\text{up}} - \eta_{\text{low}}) \cdot \frac{1}{1 + e^{k(2t-1)}} \quad (3.2)$$

In the equation, η_{up} sets the upper bounds, and η_{low} sets the lower bounds of the desired learning rate. The slope of the learning rate curve is adjusted with k [81], [11].

3.2.5. Chosen Model: DenseNet201

The results of Degtyar's thesis reveal that among the various CNN architectures analyzed, the DenseNet201 model demonstrated the best performance for classifying skin lesions. It achieved an accuracy of 0.916 and 0.835 on the validation and test datasets, respectively. This performance demonstrates the effectiveness of DenseNet201 in handling the specific task of skin lesion classification. The choice of DenseNet201 for the final version of the classification pipeline underscores its superior capability to generalize from the training data to new, unseen images. The model aims to contribute significantly to improving and accelerating the diagnostic process for skin lesions [11].

Explainability of this pipeline is not a relevant part of this thesis and therefore not addressed.

3.3. Guidelines for Software Development

The field of medical informatics has been advancing with the integration of novel technologies, particularly AI, to enhance clinical decision-making and improve patient monitoring, diagnostics, and prognostics [82].

AI can revolutionize healthcare and precision medicine [82]. The development of reusable biomedical software for research is time-consuming and likely to lack quality, therefore hindering technology transfer, reproducibility, and reusability in research communities, while software for clinical use has to be certified and cannot lack quality. To address this issue, the selected guidelines aim to make it easier for researchers to create high-quality software, improve documentation, accessibility, and traceability, and ensure reproducibility [68]. Each guideline has several activities. For overview purposes, Table 3.2 lists each of the activities described in the next sections, grouped by their respective guideline affiliation.

Table 3.2.: Academia-Tailored Guidelines and Their Activities

QMS	SLC	RM	UE
Document Management	Software Development Planning	Planning and Preparation	Use Specification
Project Planning	Requirements Analysis	Risk Policy	Use-related Risk Analysis
Project Execution	Software Architecture and Design	Iterative Risk Management Cycle	Iterative Design Cycle
Management Procedures	Implementation, Testing and Verification	Risk Monitoring	Summative Evaluation
	Software Release and Legacy Software		
	Configuration & Change Management		

3.3.1. Quality Management System Guideline

The ultimate goal of the QMS guideline is to make it easier for researchers to create high-quality software, improve documentation, accessibility, and traceability, and ensure reproducibility in order to accelerate the deployment of standardized workflows in clinical practice [68]. Some universities already have a general QMS that focuses on research and education quality, but may not address the specific needs of medical software development.

The QMS guideline specifies four procedures for documentation, which are highlighted blue in Figure 3.1. All procedures can be started at any time with no prerequisites. It is recommended to have this guideline progressed as far as possible before starting the other guidelines.

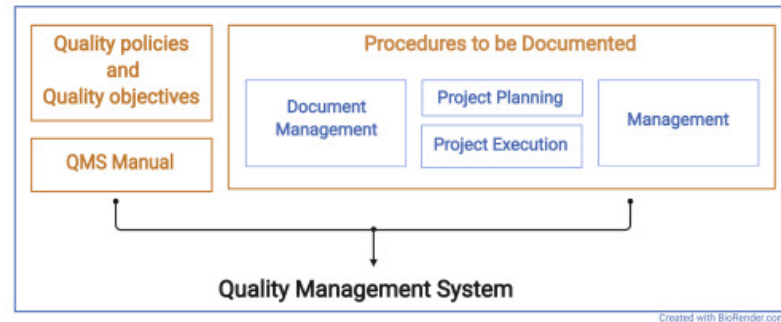


Figure 3.1.: Quality Management System Components [68]. The QMS consists of a QMS Manual, Quality Policies and Quality Objectives, and Procedures to be Documented. The guideline focuses on Document Management, Project Planning and Execution, and Management procedures.

3.3.1.1. Document Management and Management Procedures

Effective documentation is crucial for technology transfer. The first component is the Document Management, which as the name suggests, defines any management processes relating to documents, including where and how documents are stored and managed, ensuring consistency and accessibility, and any other Standard Operating Procedures (*SOPs*) that can be defined individually. It covers aspects such as document identification, access, approval, version control, and retention. The Management component on the other hand defines general management procedures, for example the procedures for QMS training to ensure that staff members are familiar with the QMS and its associated documents, or personnel qualification procedures that ensure that staff members are suitable for their roles, which may involve hiring and training [68].

3.3.1.2. Project Planning

Project Planning procedures help improve project completion and support technology transfer. This section includes procedures for choosing which QMS elements to apply, ensuring a shared understanding of project goals, defining roles and responsibilities, emphasizing risk assessment and management, and creating project-specific documents with a title, a project goal, planned start and end dates or milestones, and success and termination criteria. For each project, a document storage (e.g. Confluence) must be prepared and documented, as well as a list of all relevant created documents [68].

3.3.1.3. Project Execution

Following the Planning phase, the Project Execution phase commences. Key procedures should cover requirements management, the definition of development environments and tools, selection of development processes, criteria for project completion, and documentation levels [68].

3.3.2. Software Life Cycle Guideline

The article proposes a guideline for academic software development, tailored to research organizations' needs. It suggests a subset of elements from standard software life cycle processes that can significantly benefit research projects while maintaining feasibility, especially for academic institutions with the aim to facilitate technology transfer and improve reproducibility in a controlled and predictable manner, allowing academic advances to be deployed faster in clinical practice [69].

The main components of this guideline are highlighted in blue in Figure 3.2.

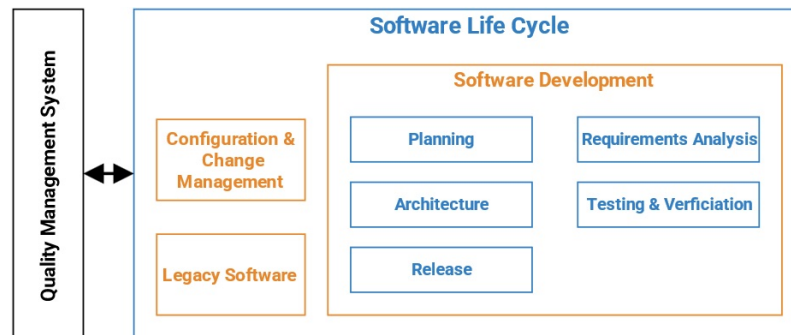


Figure 3.2.: Software Life Cycle Components [69]. A QMS is involved in the processes of the SLC, which include Configuration & Change Management, Legacy Software and Software Development. The latter has several phases: Planning, Architecture, Release, Requirements Analysis, and Testing & Verification.

The guideline for the software life cycle in medical software research encompasses various processes, including Software Development Planning, Requirements Analysis, Software Architecture, Software Design, Implementation, Testing, Verification, and Release.

3.3.2.1. Software Development Planning

In order to design and build high-quality software, one must plan a lot beforehand. This is the initial and crucial step in the software life cycle, and it includes defining a brief Intended Purpose document for the software, a Safety Class Determination based on the MDR Qualification and Classification process as seen in Fig. 1 in Appendix A.10, and a software development plan, which consists of a process description plan and a software development

plan, in the Planning phase, based on a previously selected software engineering model such as the V-model. The accurate software development plan is the first key component that must be defined and then regularly updated and referenced throughout the project. Depending on the individual software development plan, variations may occur, e.g. in the duration of tasks set. Still, all plans should roughly address used processes, deliverable results, traceability between requirements, norms, methods, risk control measures, and individually set SOPs [69]. According to the SLC guideline, it is recommended to define a coding guideline or convention, including code style, nomenclature, and naming in the software development plan, as it increases software quality [8]. It should also be one of the constraints in having a requirement marked as complete in the Definition of Done (*DoD*), which is also set in the SLC guideline. The DoD is used to check if a requirement or its related issues are fully implemented and can be tagged as completed within Confluence and Jira [8].

3.3.2.2. Requirements Analysis

A Requirements Analysis is performed after the software development plan is created, and it is updated during the project. The analysis defines software inputs, outputs and performance, but also non-functional requirements such as physical characteristics, maintenance, or cybersecurity. High-level requirements should be clearly described in a specification document, with a strong emphasis on traceability, which ensures a comprehensive understanding across various stages including design, implementation, testing, and maintenance. Furthermore, maintenance requirements should not be neglected, as maintaining software is as important as its initial development [69], [8].

3.3.2.3. Software Architecture and Design

The structure of the software is described in the architecture document together with general design decisions and the maintenance plan, covering architecture characteristics that must be supported by the system, e.g. availability, scalability, security, and architecture decisions such as rules or constraints based on the choice of the software architecture, e.g. monolithic or modular [69]. Software units are then defined together with their sub tasks as epics and issues; it is possible to delegate those into a separate detailed design document which should then also contain a mockup and a UML class diagram [69], [8].

3.3.2.4. Implementation, Testing, and Verification

Once the software units are defined, each one must be implemented, tested and verified by writing source code that follows the coding style and documentations standards that were defined earlier. The integration tests evaluate the interaction between the software

units and the system, including different levels of testing such as unit tests, integration tests, system testing, and even manual testing methods like usability tests or walkthroughs. Verification is a critical activity throughout the software development process, ensuring that all requirements are fulfilled and documented. The documents must document test results and their traceability to requirements, architecture, and design [69], [8].

3.3.2.5. Software Release and Legacy Software

Finally, academic software is typically released to other researches via public repositories or journal publications. Before release, residual anomalies must be documented and all activities defined in the software development plan must be completed, covering the entire lifetime of the medical device software. Legacy Software, which is software that was not developed for use as a medical device, such as libraries or packages, has to be assessed for possible risks and then mitigated [69].

3.3.2.6. Configuration & Change Management

The Configuration and Change Management process is crucial for usability, reproducibility, reusability, and traceability of software. In academia, often a version control system such as GitLab, GitHub, or Redmine is used. Since it is challenging in academia to implement comprehensive configuration and change management documentation, establishing technical and administrative procedures for configuration and change management, documentation, and tracking changes is essential [8]. Suggested steps in a change management process include creating problem reports, problem analysis, creating change requests, and implementing and verifying changes [69].

3.3.3. Risk Management Guideline

Transitioning medical software from academic research to industrial standards faces significant challenges due to stringent patient safety regulations. To address this, a practical Risk Management (*RM*) guideline designed for research environments is introduced, covering Risk Policy, Analysis, Evaluation, Control, and Monitoring to mitigate risks associated with software. Hazards are potential sources of harm, that when exposed to, can lead to hazardous situations. The severity is the extent of harm that can occur, and probability of occurrence is the likelihood that the harm will happen. This RM guideline is intended to streamline the transition of medical software from concept to a compliant product by mainly mitigating hazardous situations and the severity and probability of those occurring, therefore facilitating a safer and more effective integration into clinical practice [70].

The main topics of this guideline are highlighted in Figure 3.3.

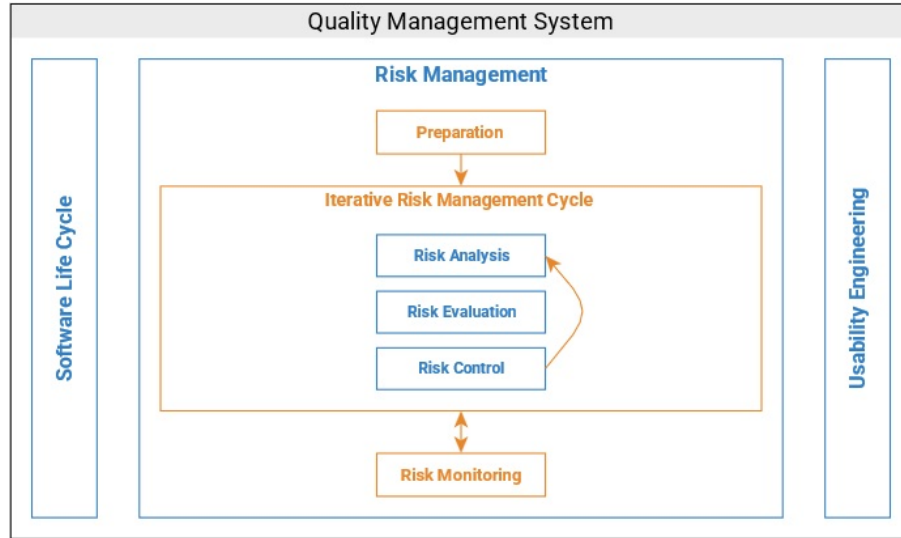


Figure 3.3.: Risk Management System Components [70]. Based on a QMS, the RM guideline is connected to both the SLC and the following UE guideline. Main components include the Preparation phase that leads to the Iterative Risk Management Cycle (*IRMC*), which consists of a Risk Analysis phase, followed by an Evaluation phase, and lastly a Control phase. After finishing the cycle, the Risk Monitoring phase commences. Depending on the results of the Monitoring phase, the IRMC might start again.

3.3.3.1. Planning and Preparation

This initial phase involves setting up the RM process, which includes planning subsequent RM activities, defining the intended use of the medical device or software, sketching a preliminary system design, and establishing a Risk Policy. The planning should be carried out early in the project to initiate all relevant documentation [83].

3.3.3.2. Risk Policy

The Risk Policy is a set of principles and criteria to determine what constitutes an acceptable risk. It is fundamental to subsequent risk decisions and is aligned with the QMS principles and quality goals. Therefore, aspects of the Risk Policy are described in detail [83]:

- **Severity Categorization:** Risks are classified by their severity into three categories: significant (death or irreversible injury), moderate (reversible injury), and negligible (no or very little damage).
- **Probability Categorization:** The likelihood of risk occurrence is also classified into three categories: high (frequent occurrence), medium (occurs a few times over the device's lifetime), and low (very rare occurrence).

-
- **Risk Matrix:** A 3x3 Risk Matrix derived from ISO 24971 is used to help researchers determine which risks are unacceptable, generally those in the upper-right area of the matrix (high probability and significant severity).
 - **Justification of Acceptance Criteria:** The researcher must justify why the Risk Matrix and acceptance criteria are suitable based on the intended use of the medical device/software.
 - **Risk Acceptance:** Criteria against which risk levels can be measured are defined. The principle is that the benefits of the application should outweigh the risks, with an emphasis on not accepting risks that could lead to death or irreversible injury.

3.3.3.3. Iterative Risk Management Cycle

IRMC 1: Risk Analysis In this phase, potential hazards associated with the software are identified, which can involve a fault tree analysis and documenting potential risks in a risk table. The risk table contains components, initial events, hazards, possible damages, and an assessment of severity and probability [83].

IRMC 2: Risk Evaluation After identifying potential risks, each risk is qualitatively estimated for severity and probability according to the Risk Policy. The Risk Evaluation involves a deliberative estimate, and a simple risk estimation and acceptance check can be conducted using a Risk Matrix, which was defined in the Risk Policy [83].

IRMC 3: Risk Control This phase explores possible risk control options, especially for risks that are not acceptable. Control measures should be documented and justified in detail. The approach is based on a priority scheme from the MDR and ISO 14971, focusing first on security and safety by design, followed by protective measures and information provision [83].

In the Appendix, a Risk Policy (Appendix A.2), the structure of the Risk Table (Appendix A.3), and the Description of the Control Measures (Appendix A.6) can be found.

3.3.3.4. Risk Monitoring

Finally, this phase ensures that the overall residual risk is acceptable. Should that not be the case, the IRMC starts again. Otherwise, the RM activities are reviewed, including an assessment of activities to date and their documentation. A RM report is produced, summarizing the RM process and including an intended use table, risk table, risk control table, and overall residual risk evaluation. The RM report should be periodically updated and forms the basis for research transfer to the industry [83].

3.3.4. Usability Engineering Guideline

The guideline focuses on the usability engineering (*UE*) process tailored for academia and research institutes, primarily within the context of developing medical software. It is derived from the IEC 62366 standard, and it includes activities like specifying the use of the software, analyzing use-related risks, designing and implementing the user interface (*UI*), and evaluating usability through both formative and summative approaches. The detailed process, as visualized in Figure 3.4, aims to ensure that the software meets the users' needs while adhering to the regulatory standards for medical devices [71].

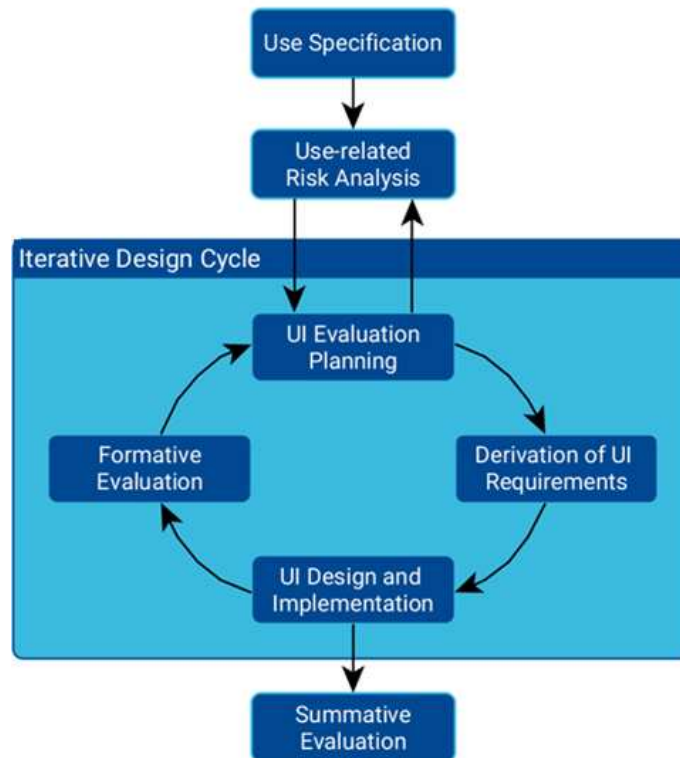


Figure 3.4.: Usability Engineering Components [71]. The flow chart starts by defining a Use Specification. Then, a Use-related Risk Analysis is performed, before starting the Usability Engineering Iterative Design Cycle (*UEIDC*). The UEIDC begins with the UI Evaluation Planning phase, followed by the Derivation of UI Requirements, then the UI Design and Implementation. Depending on the results, the UEIDC cycle either ends here, leading to the Summative Evaluation phase, or moves on to the Formative Evaluation phase. The cycle starts anew then.

3.3.4.1. Use Specification

The initial step of the UE process is to create a Use Specification document that defines the intended use of the software. It covers several key aspects [71]:

-
- **Intended Medical Indication:** Describes the medical purpose(s) of the software, e.g. diagnosis, monitoring, treatment prediction, or screening, including parts of the human body or type of tissue this software is applied to.
 - **Patient Population:** Characterizes the demographics of the patient groups who will use or be affected by the software.
 - **User Profile:** Defines the different user groups (e.g. clinicians, IT personnel) and their characteristics that could influence usability.
 - **Use Environment:** Describes the physical or social environment where the software will be used, including confounding variables such as noise, lighting, or distractions.
 - **Operating Principle:** Documents the type of inputs into the software and expected outputs.

3.3.4.2. Use-related Risk Analysis

This phase is centered on identifying potential risks associated with using the software. It involves a detailed analysis, similar to the IRMC, to uncover any use errors, hazards, and hazardous situations that might arise from interacting with the UI. The goal is to preemptively mitigate risks through design adjustments [71].

3.3.4.3. Iterative Design Cycle

The UE Iterative Design Cycle (*UEIDC*) is a core component of the usability engineering process, emphasizing the continuous refinement of the UI based on user feedback. It is divided into four phases [71].

UEIDC 1: UI Evaluation Planning This phase involves detailing the UI evaluations, planning both Formative and Summative Evaluations, their execution methods, and success criteria [71].

- **Formative Evaluation Planning:** Focuses on gathering continuous feedback during the development process to identify and fix usability issues. The plan specifies objectives, chosen methods, evaluation focus, and timing [71].
- **Summative Evaluation Planning:** Involves evaluating the final implementation of the UI to confirm that it meets the predefined acceptance criteria for usability and safety. This planning phase outlines the objective, chosen methods, UI parts to be assessed, and the criteria for successful evaluation.

UEIDC 2: Derivation of UI Requirements The second phase describes the development of detailed specifications for the UI based on the initial requirements analysis in the SLC guideline. This includes deriving UI requirements that ensure usability [71], such as:

- **UI Requirements Derived from User Needs, Preferences, and Capabilities:** Specifies requirements based on thorough understanding of the intended users' needs, capabilities, and preferences for an efficient user experience.
- **UI Requirements Associated with the Implementation of Risk Control Measures:** Identifies specific design requirements aimed at minimizing use-related risks through appropriate risk control measures.
- **Requirements for Accompanying Documentation and Training:** Outlines the need for clear and concise documentation and training materials that support safe and effective use of the software.
- **Design Principles, Heuristics, and Style Guides:** Suggests following established design principles and style guides to ensure UI consistency and usability.

UEIDC 3: UI Design and Implementation Here, the translation of UI specifications into actual design concepts with a selection of UI elements, and the subsequent implementation of these designs is covered. Prototypes (both low-fidelity and high-fidelity) should be created and used for refining the UI design based on user feedback and the Formative Evaluation results [71].

UEIDC 4: Formative Evaluation This phase describes the process of evaluating the UI during its development to identify usability issues and areas for improvement. It involves the analysis of evaluation results to determine the necessity of design changes or additional risk control measures. This component is crucial for ensuring that the the users' needs are met by the UI [71].

3.3.4.4. Summative Evaluation

The Summative Evaluation is the final evaluation of the UI upon completion of the development process. This evaluation should ideally confirm that the UI can be used safely and effectively by the intended user group in the intended use environment. It involves testing the UI against the acceptance criteria defined earlier in the Summative Evaluation Planning phase, ensuring that the software meets all usability and safety requirements [71].

3.4. Application and Implementation of the Four Guidelines

The exact description of the guideline activities is described in the sections above. It is important to note that there is no clear timeline as to when which parts of which guideline should be applied and when to switch between the guidelines since they were all written independently, which is why the decision on the practical order of implementation, as well as the implementation itself, are part of the results.

The way the guidelines are applied depends on the project structure, the resources, and other factors. In this project, the guidelines and all of their activities, as described previously, were applied successively and in parallel to one another in four stages, starting with the QMS as a central element in all guidelines. Three documentation stages existed, each dealing with different activities of all guidelines, and one programming stage, concerned with the coding and testing aspect of the skin cancer detection software.

The documentation was stored mainly on Confluence with the integration and help of a Jira project plan that was set up and updates during the project.

3.5. Software Implementation

The software implementation is equivalent to the implementation and testing part of the SLC guideline's Implementation, Testing, and Verification document.

3.5.1. Coding

The software development was planned and carried out according to the software development plan of the SLC guideline. Implementing the code for the user interfaces was achieved by using Qt Designer, an open source software for Drag-and-Drop development of GUIs. Furthermore, to write code efficiently, the PyCharm IDE was used with Python3, version 3.8, as the chosen programming language for coding and testing, and sqlite3 for the database implementation. The perks of PyCharm are listed in Appendix A.1 as part of the QMS guideline's project execution procedure.

As the project implementation progressed, changes and new functioning code lines were committed with a description message and later pushed into the Git repository to ensure traceability within the source code management.

3.5.2. Testing

Testing was performed manually by using unit tests in Arrange, Act, and Assert (*AAA*) structure and in parallel to functionality coding, following the Test-Driven-Design (*TDD*)

principle. Usability tests were also conducted. Every test was performed multiple times and was specifically designed not to interfere with the apps functionality itself, e.g. existing data in database manipulation. Only if all tests passed, another epic or issue commenced. After successfully testing and achieving a high testing coverage, the entire software was verified and documented in the Verification document as shown in the guideline implementation's Software Release and Verification step.

4. Results

In this chapter, the application and implementation of the academia-tailored guidelines is presented. Due to the fact that an activity from one guideline might need input from another, requiring a focus switch to working on that other guideline before being able to continue, this section of the thesis will be divided into several application stages, in which different activities from the guidelines are described.

Documents highly necessary for the upholding of the reading flow and the comprehension of the following results are included in the Results chapter. The longer version of some documents or documents generally too large and disruptive to the flow of reading to be added as figures in this chapter, but still crucial for the results, are attached in the Appendix and referenced.

4.1. Initial Preparation

The SLC, RM, and UE guideline all have at least one thing in common, namely the fact that a QMS is a central organizational element in all of them. It was therefore indicated to implement the QMS guideline first and progress it as far as possible before continuing with the other three guidelines. The initial preparation stage therefore focused mainly on the QMS guideline, which was also properly prepared before it was applied.

4.1.1. Setups

Selecting Confluence as the documentation platform and HHU GitLab as the code documentation platform were among the first steps taken to prepare the application of the QMS guideline. The next step was to set up and structure the Confluence-wiki, the Git-repository, and a local directory. For overview purposes, an additional Jira project plan was created, which included all tasks, their progress, and their deadlines.

Confluence Wiki

A Confluence space with the name *Software for Skin Cancer Detection* was created at first. The reason for choosing Confluence as the documentation platform was that it met the essential requirements, which were general versioning, plus versioning of renamed and deleted documents to prevent data loss, and document editing/viewing access only by authorized users, as stated in Appendix A.7 under procedure DM-2a as part of the QMS guideline.

The space served as the root directory and hosted four sub-directories, namely each of the four guidelines, a list of abbreviations, and a bibliography, as seen on the screenshot in Appendix A.5. The guideline's directories each had a landing page, further sub-directories, and sub-pages which matched their respective components.

Git and Local Repository

A Git repository with the name *Software for Skin Cancer Detection* was created by a supervisor and included only the program code for the machine learning pipeline from Degtyar [11]. After creating a local directory on a private computer for the thesis, the Git repository was pulled and saved there.

Git as a version system and GitLab as the coding documentation platform were chosen for meeting essential requirements such as code management properties, versioning, the possibility of storing intermediate states in a referenceable way, the prevention of code loss, and the possibility for simultaneous work by more than one developer on the same code, as listed in Appendix A.7 under the QMS procedure DM-2b.

Jira Project Plan

Finally, a Jira project plan was created and linked to the Confluence space. Tasks were defined and added here, and a deadline for those was set. Different levels were defined, each indicating the current status of a task being either new, in progress, finished, or overdue. Appendix A.4 shows a screenshot of the project plan.

4.2. First Documentation Stage

Considering that a documentation platform was chosen, the actual documentation of the QMS guideline in Confluence started.

4.2.1. Quality Management System Activities

The first page of the QMS guideline was the document management, in which the responsible person for the documentation was identified, the documentation platforms and their requirements were listed, the documentation life cycle and change procedure were defined, and the SOPs were documented. The Confluence page containing those procedures can be found in Appendix A.7.

Following that, the Project Planning and Execution procedures were documented. The first procedure in the Project Planning page described the project specific assignment in a table. The other two planning procedures document the steps for preparing the document storage

and a list of all necessary documents that were created during the execution of this project. The list of documents was updated throughout the project, with the final version depicted in Appendix A.8. The Project Execution document had one single procedure associated with the creation of a software development plan. A table was created and contained the choice of the development process, the description of the development environment, the description of build and integration steps, the description of requirements handling, the plan for testing, and the handling of changes. In order to fill this table, the SLC guideline had to already been started, as it contained all the relevant information for the software development plan. The choice was made to then skip this procedure temporarily. In the Jira project plan, a task was added with a reminder to return to this procedure as soon as possible.

As a final step in the documentation of the QMS guideline, the Management procedures were documented, i.e. the assignment of a responsible person for the quality management, the implementation of training for new staff, the implementation of department meetings with a quality management update, and an annual quality management review. This part was especially short as the scope of this thesis did not require any new staff member training procedures or an annual quality management review, for example.

The QMS guideline was therefore only missing the Project Execution procedure, which required a software development plan. Hence, the SLC guideline was started next, as it dealt with the creation of aforementioned plan.

4.2.2. Software Life Cycle Introduction

The recommended starting point is to state the intended purpose of the software in a document, and then to describe the process and classification used for the software's Safety Class Determination, before proceeding with the software development plan [69].

4.2.3. Intended Purpose

This document simply described the medical purpose of the app, with the goal being the support of medical professionals with the diagnosis of skin cancer in patients. A short text sufficed in describing the general procedure of how the software works, why it was developed and what needs to be known beforehand, as shown in Appendix A.9.

4.2.4. Safety Class Determination

The MDR has different requirements for different classes of software. In order to classify the software's safety class, the MDR Qualification and Classification process was described, applied, and documented next. It resulted in placing the software's safety class into the 2a category. The train of thought is attached in Appendix A.10.

4.2.5. Software Development Planning

After the initial documents were created, it was possible to begin with the Software Development Planning phase, in which a software development plan was defined. The process description plan was created first and added as a Confluence page. The scope of this software allowed for an uncomplicated process description plan, following the Linear Sequential Model, as visualized in Figure 4.1.

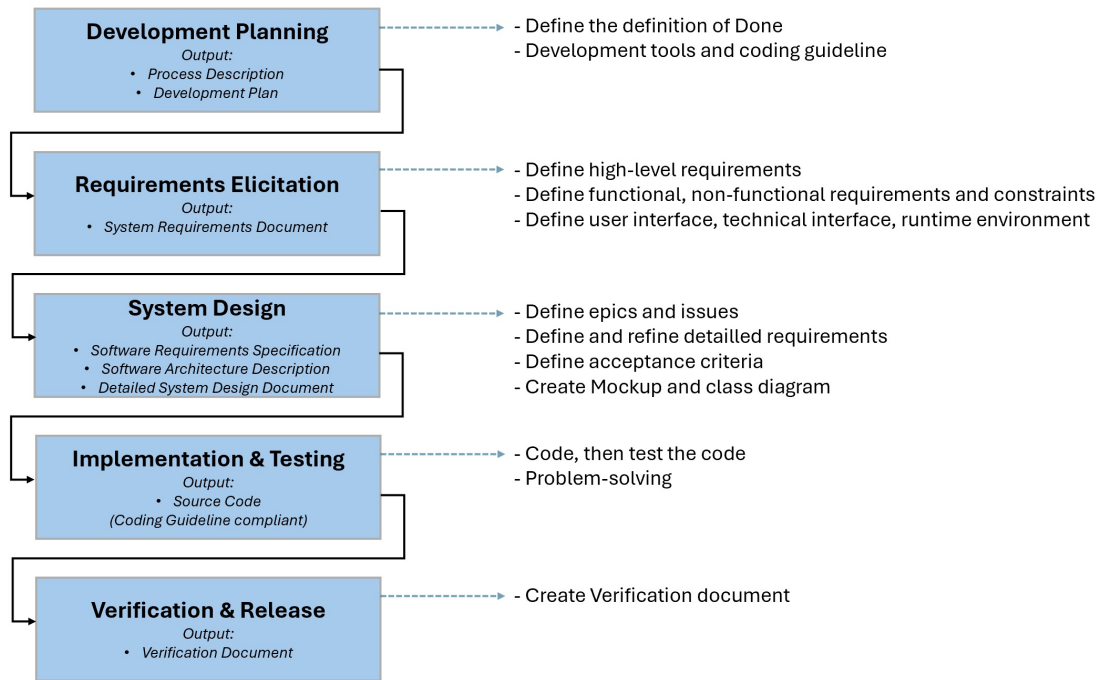


Figure 4.1.: Process Description Plan as a Linear Sequential Model. Starting at the top left, output documents are defined in the box below and the tasks associated with that phase are highlighted with a dotted arrow to the right. The next phase is always the one underneath, as displayed by the black arrows. This is similar to the Waterfall Model.

After the process description plan was finished, the actual, more detailed software development plan was drafted on its basis. It consisted of five phases and was based on the Linear Sequential Model, meaning that a phase can only begin if the previous phase has been finished successfully. Additionally, the definition of done and the coding guideline both were drafted in the Software Development Planning page. The software development plan's document is depicted in Figure 4.2 and again with the DoD and coding guideline in Appendix A.11.

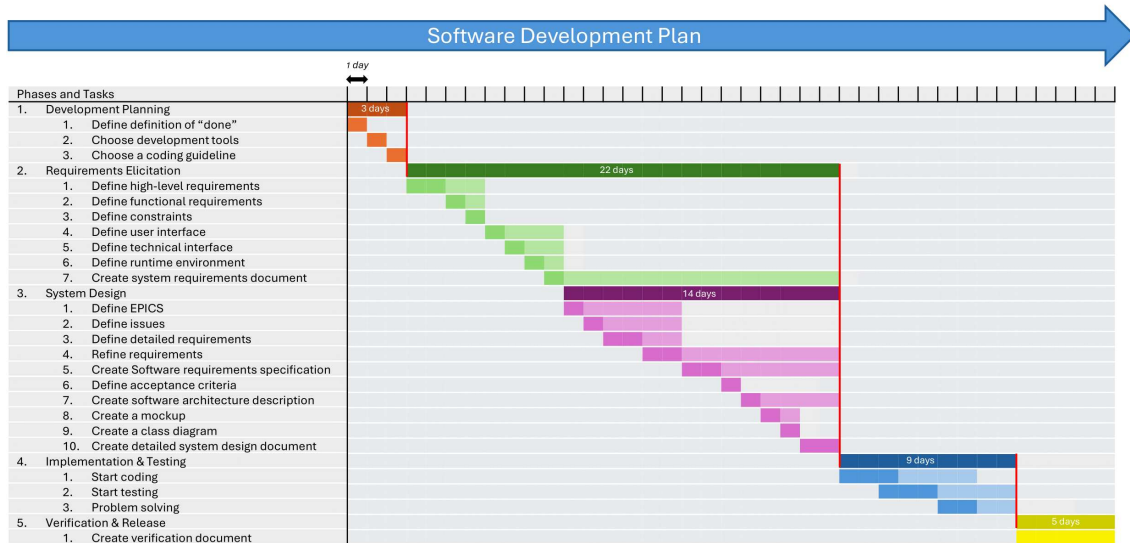


Figure 4.2.: Software Development Plan. Phases and their tasks are on the left, with their respective duration in the Gantt-chart on the right. Some tasks have lighter colored bars, indicating the maximum additional days that the estimated duration of those tasks may exceed. Red lines symbolize the fact that all phases, except two and three, may only commence when the phase at the top of the line is finished in its entirety.

Following the software development plan, the rest of the documents needed for the SLC guideline compliance should have been created, starting with the Requirements Elicitation phase, which was followed by the System Design phase, then the Implementation & Testing phase, and lastly, the Verification & Release phase. However, for those documents, several activities from the UE and the RM guideline must have started already. The residual tasks regarding the documents mentioned for the SLC guideline were therefore updated in the Jira project plan with a note to return to those as soon as base preparations for the UE and RM guideline were conducted.

4.2.6. Use Specification

Defining a more detailed version of the Intended Purpose document of the SLC guideline, a Use Specification, was the initial step in applying usability engineering. It detailed [71]:

- The intended medical indication.
- The patient population.
- Parts of the human body or type of tissue applied to.
- The user profile.
- The use environment.
- The operating principle.

This document was finished in its entirety, as shown in the screenshot in the Appendix A.12, before proceeding to start the Iterative Design Cycle (*UEIDC*) of the Usability Engineering guideline.

4.2.7. Iterative Design Cycle Initialization

The first UEIDC started off with the first phase, the User Interface Evaluation Planning, in which the contents of the Formative and Summative Evaluation plans were defined with the chosen evaluation method of observing and inquiring representative users, e.g. with usability tests. Then, as described in the second phase of the UEIDC, user interface requirements are derived. This step has been delegated to be documented in the System and Software Requirements Specification, a document introduced later in the SLC guideline, for overview purposes. Additionally, only after at least having a draft of the Use Specification document, the Use-related Risk Analysis could start. This led to the preparation of adapting the RM guideline.

4.2.8. Risk Management Planning and Risk Policy

In order to apply risk management to the project, a planning document was drafted, which included the location of the RM documents and a table depicting the schedule of planned RM activities such as the risk analysis phase, evaluation phase, control phase, initial monitoring, and monitoring intervals, together with the identity of the person who was responsible for the procedures. The document also contained the intended purpose table, derived from the SLC guideline's Intended Purpose document and the UE guideline's Use Specification. It is attached in Appendix A.13. The next step here was to create the Risk Policy, which contained a Risk Matrix, defining what risks are acceptable or unacceptable, based on the three-class severity and probability classifications proposed in the guideline. The Risk Policy's Risk Matrix and the severity classes are depicted in Figures 4.3 and 4.4.

Severity Level of Harm	Description
Negligible	Results in inconvenience or temporary discomfort, no injury
Moderate	Results in injury or impairment requiring or not professional medical care
Significant	Results in permanent impairment or life-threatening injury

Figure 4.3.: Extract of the Risk Policy: The Severity Classes. They are either negligible (no damage/very little damage), moderate (reversible injury occurs), or significant (death or irreversible injury).

	Negligible	Moderate	Significant
High			
Medium			
Low			

Figure 4.4.: Extract of the Risk Policy: The Resulting Risk Matrix. Dark-grayed out cells are unacceptable risks, while white cells are acceptable. The probability of the occurrence is categorized as either high (occurs frequently), medium (occurs a few times during the lifetime of the SaMD), or low (occurs very rarely).

As the software is only a supportive instrument, it was determined that it is very unlikely that risks exist that have a high frequency and are significant or moderate, or medium frequent risks with a moderate/significant severity.

4.3. Second Documentation Stage

After the fundamental, preparing activities of the UE and RM guideline were done, the second documentation stage began. In this stage, mainly the SLC and UE guidelines were the focus.

4.3.1. System and Software Requirements Specification

This document of the SLC guideline consisted of all task outputs for phase two, the Requirements Elicitation, of the software development plan in Figure 4.2. Since this software was standalone, it was not possible to clearly distinguish between system and software requirements [69], as such, they were combined. The content of this document included, amongst others, a table of high-level functional requirements as seen in Figure 4.5.

ID	Requirement	User Story / System Requirement	Priority
1	High-level requirement	As a user, I want to get the different values for the skin lesion classifications.	Necessary
2	High-level requirement	As a user, I want to know what the risk scores indicate and what consequences arise.	Necessary
3	High-level requirement	As a user, I want to be able to capture skin images with an external tool.	Necessary
4	High-level requirement	The software should be up and running within 2 minutes.	Necessary

Figure 4.5.: High-Level Functional Requirements from the System and Software Requirements Specification.

Moreover, a list of other functional requirements, a table of non-functional requirements, a list of constraints, a technical interface description, a runtime environment, and the UI requirements from the UE guideline were added. These specific collected requirements can be found in Appendix A.14.

4.3.2. System and Software Documents

Following the software development plan, in the System and Software Design document, the acceptance criteria as well as epics were defined. The acceptance criteria are as follows:

1. All unit tests pass.
2. Test coverage of at least 50 %.
3. Risk score calculations are reproducible.
4. The stakeholder is satisfied.

The epics included the app icon and each screen in the app, namely:

1. The welcome page.
2. The data input page.
3. The user history page.
4. The image preview page.

Additionally, each of those pages had their sub-tasks or elements (issues) listed below. For the welcome page, those were the app layout, the welcome text, and the process start

button. The issues in the data input page included the user information class, user data entries, and three buttons for navigation and importing user data. In the history page epics, the issues were the back navigation button, the new image button, and the image and results history preview. Finally, the image preview page covered the results chart and the implementation of a restart, a capture image, and a save results & image button. The app icon epic deals with the creation of the app icon. The complete design document is found in Appendix A.15.

Next, the structure of the software was described in the Software Architecture Description document, including general design decisions such font sizes for example, an app icon, the screen and window designs, software maintenance, and the detailed design of each page, as depicted in Appendix A.16.

Finally, in Appendix A.17 the Configuration and Change Management document was created.

4.3.3. Iterative Design Cycle Continuation

Designing the user interface and implementing it was the next step before proceeding with the actual coding. The third phase of the UEIDC discussed the choice of UI elements, e.g. buttons for navigating through the app, the placement of the GUI elements, visual details, the interaction design, and the versions of the user interface. Knowing how the architecture and the software design were supposed to look like, it was possible to create prototypes of the UI with different tools such as figma or wireframes.cc. Here, one version was implemented first, however, when proceeded with the UEIDC phase four, the Formative Evaluation phase, changes had to be made. Ultimately, the UEIDC had to start anew, until after three times, a viable version was created, which also served as a mockup in the SLC guideline's System and Software Design document. The documentation of phases 3 and 4 of the UEIDC can be found in Appendix A.18.

4.3.4. Mockup and Class Diagram

Eventually, after creating wireframes in the UI Design phase in the UEIDC phase three, a basis for a more detailed mockup was given. The class diagram was created immediately after. The mockup process can be viewed in Appendix A.19 and the class diagram, due to its large size and small font rotated, in Appendix A.20. The software was now ready to be coded and the QMS guideline's Project Execution table, as reminded by the Jira project plan, has been documented. Note that for the flow of this thesis, the coding process which was a part of the SLC guideline's Implementation, Testing, and Verification document, follows after the final documentation stage.

4.4. Final Documentation Stage

After the coding of the software, the only procedures left to be documented were the IRMC of the RM guideline including the Risk Monitoring, the Summative Evaluation and the Use-related Risk Analysis of the UE guideline, and the Software Release and Verification document of the SLC guideline.

4.4.1. Use-related Risk Analysis

The Use-related Risk Analysis, as depicted in Appendix A.21, contained a risk sheet of a collection of potential use errors, hazards, and situations related to usability. It started off with the identification of safety-related UI characteristics in a table with potential use errors, then an analysis of all possible consequences triggered by human errors was performed, before identifying hazard-related use scenarios. In the end, risk control measures were listed, according to the risk control types from Appendix A.6. The risks are included in the IRMC of the RM guideline in the next sections.

4.4.2. Summative Evaluation

The Summative Evaluation then consisted of, additionally to its base contents, a final list of hazard-related use scenarios. It also contained the usability test protocol, with exemplary usability tests and their descriptive evaluation. One of those tests is depicted in Figure 4.6. As a key document, the entire Summative Evaluation is added in Appendix A.22.

Test ID	Test	Hazard-related use scenario	Evaluation
T1	View entry history of user with the name "Tester, TestUser".	U2	<p>The user's task was to show us the history of the user with the name "Tester, TestUser". He was not provided with any additional material except the running software itself.</p> <p>The user went on to read the introductory text and use instructions on the welcome screen. He then clicked on the "Start"-button and landed on the data input screen. There, he read the informative text and clicked on the "Import data"-button; shortly after, the dialog input field opened, requesting a full name. After inputting the first letters of the user name: "teste", the software autorecommended the name "Tester, TestUser" and the user confirmed by pressing Enter. He clicked on the "Ok"-button, leading to the closing of the dialog input window and the importing of the user data. After double checking the first- and last name imports in the line edit fields, the user clicked on the "Proceed"-button and was able to view the entry history of the user.</p> <p>The second user was provided with the same material and performed the task identically with one small difference: Instead of inputting the first letters of the user name into the dialog input window, the user clicked on the drop-down menu and simply chose the user "Tester, TestUser".</p> <p>No risks, errors or difficulties were observed.</p>

Figure 4.6.: Extract from the Usability Test Protocol Table of the Summative Evaluation.

4.4.3. Iterative Risk Management Cycle

The results of the Summative Evaluation were transferred to the IRMC and then it was judged whether the overall residual risk of the SaMD was acceptable or not [83], [72]. In this project, only one IRMC had to be performed. In the first phase, a hazard analysis table identified four hazards, which are illustrated in Figure 4.7, and eight risks as seen in Figure 4.8, which were assessed in the Evaluation phase afterwards according to the Risk Policy.

ID	Hazard	Hazardous situation
H1	Patient <i>health harm</i> due to no treatment	SaMD output
H2	Patient <i>health harm</i> due to incorrect treatment	SaMD output
H3	Manipulation of other <i>systems</i> by faulty information	Any kind of faulty output can lead to faulty states in other systems (unexpected behavior)
H4	Violation of <i>data</i> protection, privacy, and intellectual property	Incorrect output data and faulty requests can cause other systems to manipulate, process, or delete personal data as part of system manipulation, unauthorized access

Figure 4.7.: IRMC 1: Hazard Analysis. Four possible hazards were identified. The H1 hazard is that a patient may receive no treatment, as perhaps incorrectly suggested by a result of the SaMD output. Similarly, for H2 a patient may receive a wrong treatment as a result of the SaMD output. The hazard H3 is that other systems might get a faulty SaMD output as an input and therefore deliver unexpected results. Lastly, the analyzed hazard H4 is that sensitive patient data may be involuntarily disclosed, deleted or manipulated.

Risk ID	Component	Error (Cause)	Hazard	Damage
R1	Database	Database not reachable or takes too long to answer	H1	No (or delayed) treatment
R2	Application	Incomplete form submission (missing inputs)	H1	No (or delayed) treatment
R3	Database Security	Malicious SQL injection	H1, H2, H3, H4	No treatment, false treatment, system manipulation or user data violation
R4	Database Security	Unauthorized access to stored patient diagnoses	H4	Privacy violation
R5	Database Security	Malicious user floods system with fake data	H3	System damage
R6	Application	User provides wrong name	H1, H2	No or false treatment
R7	Application	User uploads garbage data (e.g. unrelated images)	H1, H2	No or false treatment
R8	Communication	User provides wrong contact info	H1, H2, H4	No treatment, false treatment or privacy violation

Figure 4.8.: IRMC 1: Risk Analysis. The table is exemplary and not exhaustive.

A screenshot of the Risk Evaluation table is depicted in Figure 4.9. It is important to note that the Risk Evaluation also included UE risks from the Use-related Risk Analysis of the UE guideline.

ID	Error (Cause)	Probability	Severity
R1	Database not reachable or takes too long to answer	Low	Negligible
R2	Incomplete form submission (missing inputs)	High	Significant
R3	Malicious SQL injection	Medium	Significant
R4	Unauthorized access to stored patient diagnoses	Low	Significant
R5	Malicious user floods system with fake data	Low	Significant
R6	User provides wrong name	Medium	Moderate
R7	User uploads garbage data (e.g. unrelated images)	Low	Significant
R8	User provides wrong contact info	Medium	Negligible

Figure 4.9.: IRMC 2: Risk Evaluation. The table includes use-related risks.

Figure 4.10 displays the Risk Matrix that was filled, indicating the occurrence of three unacceptable risks.

	Negligible	Moderate	Significant
High	0	0	1
Medium	1	1	1
Low	1	0	3

Figure 4.10.: IRMC 2: Risk Matrix. Following the matrix, three unacceptable risks were identified: R2, R3, and R6.

The risks were subsequently controlled in the Risk Control phase later as seen in Figures 4.11 and 4.12, mostly with the inherent safety-by-design control measure.

ID	Control Measure	Description
R2	Inherent safety by design	The user is not allowed to proceed unless all required input fields are filled.
R3	Inherent safety by design	The code is programmed in a way that does not allow SQL injections. This is achieved by implementing parameterized SQL queries and not taking the input directly and putting it into a SQL statement.
R6	Inherent safety by design and information for safety	The user must double check entries and is reminded of that by the informative text showing on the respective page. Additionally, if a user wants to import a user and he inputs a wrong name, an error window pops up, forbidding that action and preventing the risk.

Figure 4.11.: IRMC 3: Not Acceptable Risks and the Description of Their Control Measure.

ID	Control Measure description
R1	Efficient programming and implementation of load balancing
R4	Encryption of stored user data, implementation of access logs, local servers
R5	Implementation of firewalls and monitoring systems, local servers
R7	Adding clear instructions and recommendations for uploading data
R8	Implement email-verification procedure or similar

Figure 4.12.: IRMC 3: Acceptable Risks and the Description of Their Control Measure. Control measures are not mandatory to implement as those risks are acceptable, it is however favorable.

The project success had to be evaluated here and it was concluded that no danger is imminent as there are no highly critical and not controllable risks making the project

continuation impossible. Finally, the IRMC concluded with the Risk Monitoring document, which showed the updated Risk Table and Risk Matrix after controlling the unacceptable risks, as seen in Figures 4.13 and 4.14.

Risk ID	Component	Error (Cause)	Hazard	Damage	Implemented?
R1	Database	Database not reachable or takes too long to answer	H1	No (or delayed) treatment	No
R2	Application	Incomplete form submission (missing inputs)	H1	No (or delayed) treatment	Yes
R3	Database Security	Malicious SQL injection	H1, H2, H3, H4	No treatment, false treatment, system manipulation or user data violation	Yes
R4	Database Security	Unauthorized access to stored patient diagnoses	H4	Privacy violation	No
R5	Database Security	Malicious user floods system with fake data	H3	System damage	No/Obsolete
R6	Application	User provides wrong name	H1, H2	No or false treatment	Yes
R7	Application	User uploads garbage data (e.g. unrelated images)	H1, H2	No or false treatment	Partially
R8	Communication	User provides wrong contact info	H1, H2, H4	No treatment, false treatment or privacy violation	No/Obsolete

Figure 4.13.: Risk Monitoring: Final Updated Risk Table. Stricken entries no longer pose a risk as they have been mitigated.

	Negligible	Moderate	Significant
High	0	0	0
Medium	0	0	0
Low	1	0	2

Figure 4.14.: Risk Monitoring: Updated Risk Matrix. No unacceptable risks exist anymore.

No residual risks were found and as such, the project was finished from the RM perspective.

4.4.4. Verification and Release

In the last step in adapting the guidelines, the Verification document and the Software Release document of the SLC guideline were created. The Verification document was part of the large Implementation, Testing, and Verification document, and explained how the DoD, as defined early on in the development plan, was used as a scheme to tag an epic or an issue as finished, and how the coding guideline specified in that same document was also individually screened for. The verification of all epics, issues and above listed implementations was given by testing rigorously and achieving a testing code coverage of 73 % for the entire project structure, which surpassed the 50 %-minimum required coverage set in the acceptance criteria.

Finally, since the software development plan was finished and the code was compliant with the DoD, the coding guideline, the Requirements Specification, the project architecture, and the detailed design, as well as other relevant documents, licenses, and third-party libraries, the app was complete from the developmental point of view.

In order to release the software, an interactive checklist in the Software Release document was assessed and accepted. It included the following points:

- All known rest anomalies have to be documented and evaluated.
- Documentation of the procedure and the software development environment have to be recorded as well as the version of the released software.
- All activities and tasks of the software development plan must be completed and documented.
- The medical device software, all configuration elements, and the documentation have to be filed for the whole lifetime of the medical device software.

This simple checklist was the final procedure, meaning that all activities of the QMS, SLC, RM, and UE guidelines were now fully implemented. The workflow chart in Figure 4.15 visualizes the different application stages including the guideline focus switches.

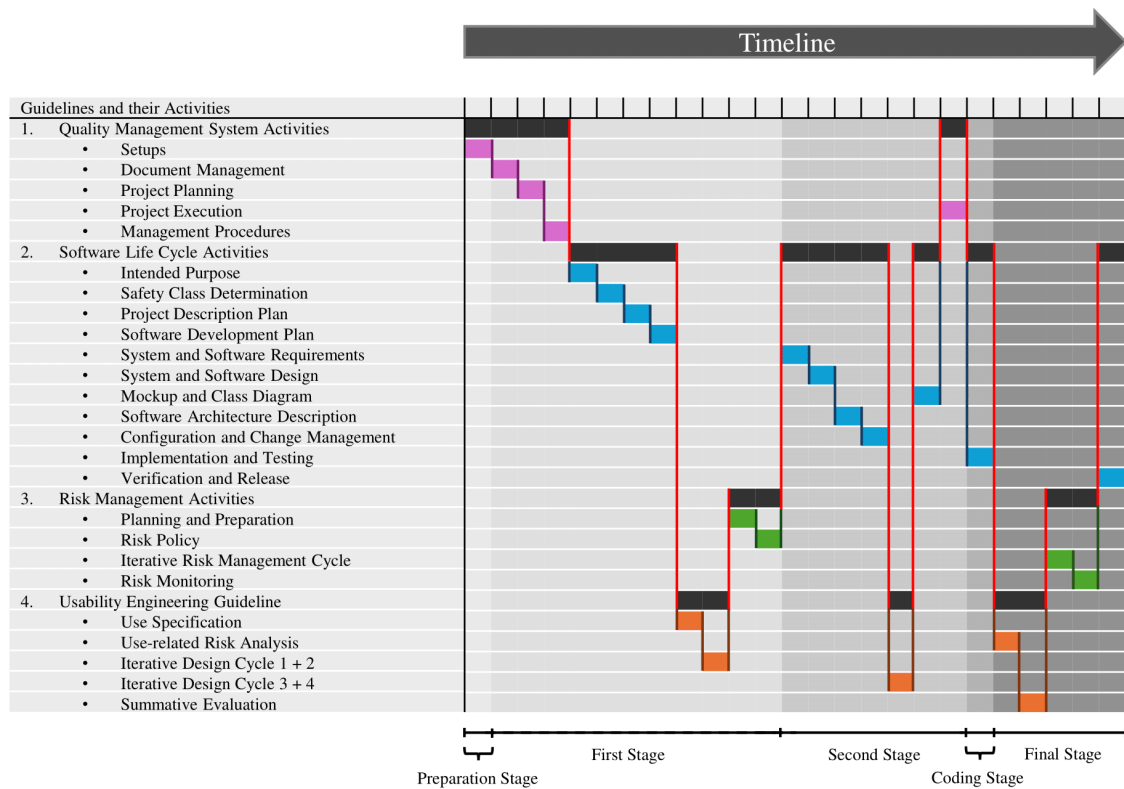


Figure 4.15.: Visualization of the Guideline Applications in Relation to the Aforementioned Stages. On the left, the guidelines and their activities are listed. On the right, the timeline from left to right is depicted, with purple blocks representing the QMS guideline, blue the SLC, green the RM, and orange the UE guideline. The black blocks illustrate which guideline is being focused on at the time, with the connecting red lines visualizing the jumps between the guidelines. Lastly, on the bottom of the chart, the initial preparation stage, the first, second, and final documentation stage, as well as the coding stage are depicted, each having a darker grey tone in the graph for better stage allocation and overview.

4.5. Software Implementation

This section describes the programming and testing process of the software for skin cancer detection. The source code is available on GitLab, including a README file attached in Appendix A.24 with a brief instructive description and installation remarks.

4.5.1. Coding

Firstly, as mentioned before, the app icon and the following user interfaces were defined as epics: the welcome page, data input page, history page, and image preview page. Their issues were also added.

Following this step, the basis was given for implementing the code for those interfaces with Qt Designer. Having created the interfaces' .ui-files, the local folder and the GitLab repository, which existed already thanks to the QMS guideline's Document Management process, were updated to include these files.

The final layout of the screens is depicted in the final version in the UI Design and Implementation step of the UEIDC, found in Appendix A.18.

After embedding the .ui-files into the project, the prerequisites for implementing functionality were given. The buttons which connected all screens were implemented afterwards, allowing bi-directional navigation identically to the depiction of the mockup process in Appendix A.19, and the introduction of app logic:

- The "Start"-button on the welcome screen (state 0) leads to the data input screen (state 1).
- The "Back"-button on the data input screen leads back to the welcome screen.
- The "Proceed"-button on the data input screen leads to the history screen (state 2), or if it is a new user, to the image preview screen (state 3).
- The "Back"-button on the history screen leads back to the data input screen.
- The "New image"-button on the history screen leads to the image preview screen.
- The "Restart"-button on the image preview screen leads back to the welcome screen.

Having created the basic navigation logic, a database with two tables was created next. The first table *users* stored all user information, including a clearly identifiable user ID, the first and last name, the birth date, an additional information text, and contact information such as the home address, email address, and phone number. The attributes, their type, and purpose can be found in Table 4.1.

Table 4.1.: Database: Table *users* Attributes.

Attribute Name	Attribute Type	Required?	Purpose
id	Integer, primary key	Yes	Used to clearly identify a user.
firstName	Text	Yes	Stores user's first name.
lastName	Text	Yes	Stores user's last name.
birthdate	Text	Yes	Stores user's birth date.
homeAddress	Text	No	Stores user's home address.
email	Text	No	Stores user's contact email address.
phoneNumber	Text	No	Stores user's contact phone number.
additionalInfo	Text	No	Stores any additional notes.

The second table *imagesAndResults* saved details of images captured with the dermatoscope, as well as the calculated results of the machine learning pipeline. All attributes, their type, and their purpose can be found in the following Table 4.2.

Table 4.2.: Database: Table *imagesAndResults* Attributes.

Attribute Name	Attribute Type	Required?	Purpose
id	Integer, primary key	Yes	Used to clearly identify an image.
dateTaken	Integer	Yes	Stores the date of the taken image.
imageName	Text	Yes	Saves the local name of the image.
physicianEntry	Text	No	Stores the physician's assessment (0 = none, 1 = akiec, ...)
resultAkiec	Integer	Yes	Stores the calculated akiec risk score.
resultBcc	Integer	Yes	Stores the calculated bcc risk score.
resultBkl	Integer	Yes	Stores the calculated bkl risk score.
resultDf	Integer	Yes	Stores the calculated df risk score.
resultMel	Integer	Yes	Stores the calculated mel risk score.
resultNv	Integer	Yes	Stores the calculated nv risk score.
resultVasc	Integer	Yes	Stores the calculated vasc risk score.
belongsToUserId	Integer, foreign key	Yes	Stores the userId from the user it belongs to from the <i>users</i> table.

Subsequently, implementations were coded for the data input screen, including:

- The line edit fields on the data input screen were activated and inputs into them are saved in variables if "Proceed"-button is clicked.
- The "Proceed"-button now requires at least three inputs: the first name and last name line edits, and the birthday date edit; if one input is missing, an 'invalid input' error window pops up and the button will not change the logic to show the history (or image preview) screen.

Following that step, database connections were established among other implementations, including:

-
- Data input screen:
 - Create a new user in the database if a user with the inputted first name, last name and birthday does not exist.
 - Update a user if a user that exists by cross referencing first name, last name and birthday has a new entry, e.g. new phone number, or wants an optional entry removed, e.g. home address.
 - Import a user if "Import data"-button is pressed by inputting a userId in an external dialog window.
 - History screen:
 - List all previous user entries/images by selecting them from the database.

Embedding the machine learning pipeline that calculates risk scores and finalizing it was divided into the following tasks:

- Defining the camera.
- Starting the camera when the "New image"-button on the history page is clicked, or when the "Proceed"-button on data input screen is clicked and the user is new and has no history.
- Implementing a new "Capture"-button that captures an image and saves it temporarily in a sub-folder.
- Sending that captured image through the ML pipeline and displaying classification risk scores.
- Adding functionality to the "Save"-button, that saves an image including risk score results to the database, links it to the user, and saves images and results as independent files locally.
- Eliminating a risk that could result in storage-disorganization by having captured images that were not explicitly saved by clicking on the "Save"-button be overwritten if the "Capture"-button was pressed again.
- Eliminating a saving error by disabling the "Save"-button until an image was captured.
- Closing the camera when the "Restart"-button is pressed.
- Resetting results and deleting non-saved captured images if app is closed or "Restart"-button is pressed.

The software requirements were updated which resulted implementing the additional following functions:

-
- The "Import data"-button on the data input screen now asks for a full name in the "last name, first name" format instead of asking for a userId.
 - The physician entry column on the history table in the history screen now allows a physician to choose a value from a drop-down menu and the system updates that value consequently in the database.
 - The images listed in the history table of a user in the history screen can now be clicked on and previewed.

Finally, after the Summative Evaluation process, all other set requirements were double checked and verified:

- All high-level functional requirements are fulfilled.
- All other functional requirements are fulfilled.
- All non-functional requirements are fulfilled.
- All constraints are checked.
- All UI requirements are fulfilled.

4.5.2. Testing

Two different people were invited to test the software and report any noticed mishaps. Fortunately, during usability testing, only two typos were discovered and one input error, which were subsequently corrected. The usability test protocols are described in Appendix A.22.

This is the list of all tests conducted and a description for each. How each test worked is documented in the Git source code in commentary form. All tests passed.

1. Test for existence of the app.
2. Test if app switches to welcome screen if "Back"-button is clicked on data input screen.
3. Test if app switches to data input screen when "Start"-button is clicked on welcome screen.
4. Test if app switches to data input screen if "Back"-button is clicked on history screen.
5. Test if app stays on data input screen if "Proceed"-button is clicked when not all required fields have been filled.
6. Test if app switches to history screen if "Proceed"-button is clicked when at least all required fields have been filled.
7. Test if app updates the user data in database if "Proceed"-button is clicked.

-
8. Test if app creates a new user in database if "Proceed"-button is clicked and user does not already exist.
 9. Test if app switches to image preview screen instead of history screen if "Proceed"-button is clicked and the input is valid with a new user.
 10. Test if app retrieves correct user from database when "Import data"-button is clicked.
 11. Test if app imports no user and gives error if "Import data"-button is clicked but given name is not valid.
 12. Test if app opens the image when clicked on the cell with image name in history table.
 13. Test if app does not open anything if any other cell is clicked except the image name cell.
 14. Test if app changes physician entry in database when drop-down menu choice is changed in history table.
 15. Test if app disables "Save"-button when switching to image preview page.
 16. Test if results get reset or show n/a when switching to image preview page.
 17. Test if app switches to welcome page if "Restart"-button on image preview page is clicked.
 18. Test if app saves a captured image temporarily when "Capture"-button is clicked.
 19. Test if app deletes a captured temporary image that was not saved when app is closed.
 20. Test if temporary captured image gets updated with a new temporary image with the same name and id if "Capture"-button is clicked and image was not saved.
 21. Test if "Save"-button gets enabled when "Capture"-button is clicked.
 22. Test if app switches to image preview page when "New image"-button is clicked on history page.
 23. Test if app saves the image locally when "Save"-button is clicked on image preview page.
 24. Test if app closes the camera when the "Restart"-button is clicked on image preview page.
 25. Test if app closes the camera when app is exited.
 26. Test if birthday gets decoded correctly for database input.

4.6. Final Software Design and Documents

The final version of the app is depicted in Figures 4.16 to 4.19.

Software for Skin Cancer Detection

Welcome!

Thank you for using this application.
Please read the software use specification thoroughly before proceeding.

To use this software, you must be a licensed physician.
Please note that the results of this software are only to be used in support of a diagnosis.
It is in no way a substitute for a clinical examination.

Use instructions

Please make sure a dermatoscope is connected.
Double-check any data entered, images taken and results presented.

Here is a general walkthrough.

1. Enter user data on the next page, or import data using the ID of the user.
2. View previous images and results on the page after if user existed.
3. Capture a new image on the next page by pressing on the capture-button.
4. Results are presented when an image is captured.
5. Make sure you save the results. Multiple pictures can be taken by pressing the capture-button and their results are presented, but only saved results will be stored in the database.
6. Restart the process or exit anytime by clicking on 'X' on the top right.

Start

For questions and feedback contact: msha104@uni-duesseldorf.de

Figure 4.16.: Screenshot of the Welcome Page of the Software for Skin Cancer Detection.

Software for Skin Cancer Detection

User information

Please input new user info below, or import an existing user. Please double check entries before proceeding.
Contact information and additional notes will be updated if new info is provided or old info is removed.

First Name* Birthdate*

Last Name*

Contact information

Home address

E-mail address

Phone number

Additional notes

Back **Import data** **Proceed**

For questions and feedback contact: msha104@uni-duesseldorf.de

Figure 4.17.: Screenshot of the User Data Input Page of the Software for Skin Cancer Detection.

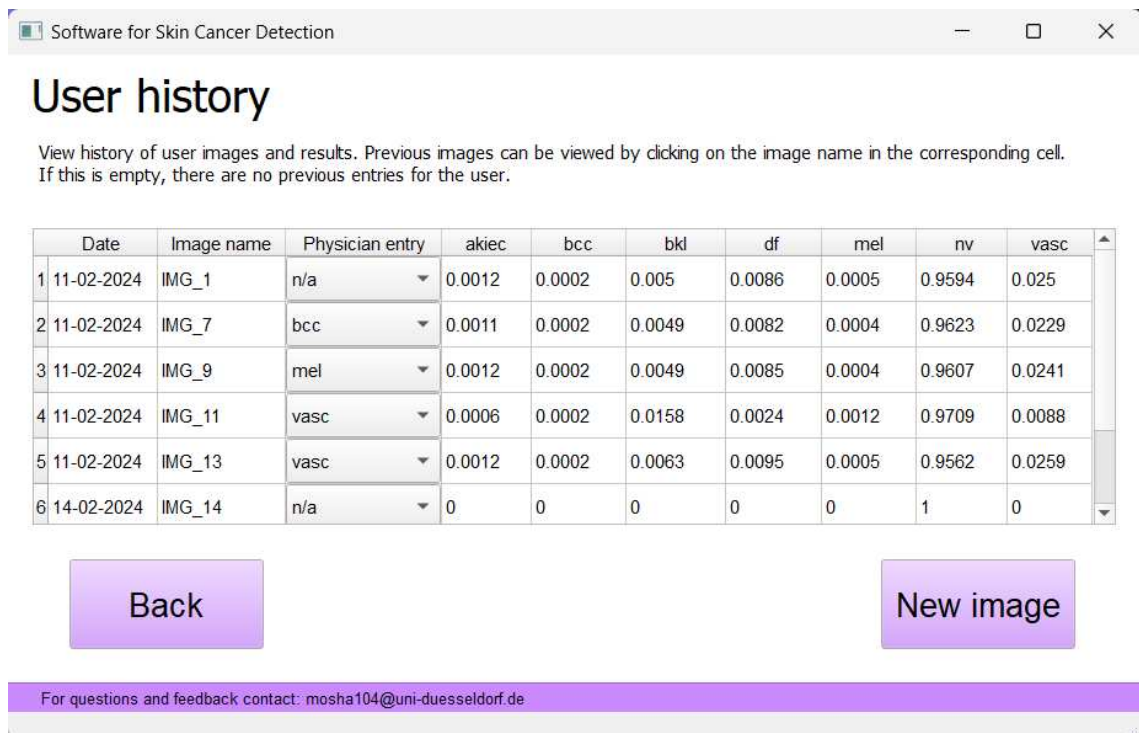


Figure 4.18.: Screenshot of the User Entry History Page of the Software for Skin Cancer Detection.

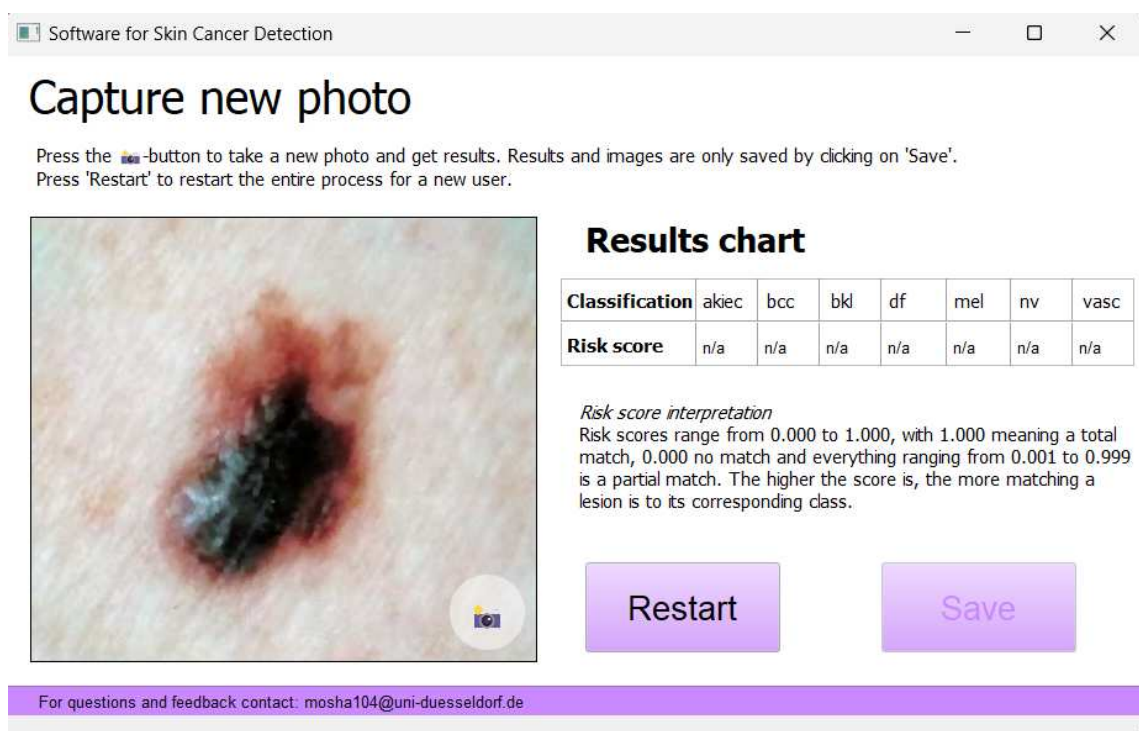


Figure 4.19.: Screenshot of the Image Capturing Page of the Software for Skin Cancer Detection, with a lesion in the camera preview.

All relevant documents as collected in the list of documents during the QMS guideline's Project Planning phase are either attached in the Appendix or added as figures. They include the Safety Class Determination (Appendix A.10), the Intended Purpose document (Appendix A.9), the Software Development Plan consisting of the Process Description Plan and the Development Plan (Figures 4.1 and 4.2), the System and Software Requirements Specification (Appendix A.14), the Software Architecture Description (Appendix A.16), the System and Software Design document (Appendix A.15), the Mockup (Appendix A.19 and Class Diagram A.20), the Software Problem-Solving Process (Appendix A.25), the Risk Policy (Appendix A.2), the Risk Monitoring document (Appendix A.23), the Use Specification (Appendix A.12), and the Summative Evaluation (Appendix A.22).

5. Discussion

The core objective of this thesis was to develop a clinical decision support system aimed at enhancing the accuracy and efficiency of skin cancer detection by integrating four academia-tailored guidelines, documenting their processes thoroughly. To conclude, these guidelines collectively contributed to a robust development framework, culminating in software close to regulatory standards. Leveraging the machine learning classification pipeline [11], the final software is able to analyze dermatoscopic images and classify skin lesions into seven prevalent categories.

The application of the Quality Management System guideline [68] ensured rigorous documentation and management processes, while the Software Life Cycle guideline [69] application facilitated a systematic development process, from planning to implementing. The Risk Management guideline [70] helped with identifying and mitigating potential risks associated with software usage, prioritizing patient safety and data security. Centered on optimizing the user interface and experience, the software was iteratively refined during the application of the Usability Engineering guideline [71].

Although these four guidelines were implemented completely and successfully, the process of applying them faced a few challenges, possibly due to the independent yet intertwined nature of the guidelines, including redundancy of activities or ambiguity. For example, the intended purpose statement, while primarily referenced and discussed in the SLC guideline, also appears in the RM guideline, which expects identical content. Another example is the location of saved documents. The QMS guideline and the RM guideline both specify where documents are stored and who has access to them, leading to potential redundancy. This is perhaps not a major issue in this project with a clear scope, but possibly in projects with a larger, more complex structure. Suppose a project is fully developed using other guidelines and is only missing a risk management component. Then, the RM guideline is used exclusively in an independent manner, possibly even supplementing other guidelines should they be missing an Intended Purpose document.

Another challenge in applying the guidelines was the fact that there is no clear timeline for when activities from different guidelines should start. Internally, each guideline has a comprehensive timeline for its own activities, but not for activities requiring input from another guideline, such as the Requirements Specification of the SLC guideline, which requires user interface requirements stemming from the Iterative Design Cycle phase two of the UE guideline. These recursively require results from phase one of the UEIDC, leading to unpredictable and costly switches in applying those guidelines. A clear interconnected timeline or a fifth supplementary guideline that assists in applying and interpreting the academia-tailored guidelines could be beneficial, similar to the international supplementary ISO 24971 standard which assists in implementing the ISO 14971.

One should further note that the software development plan employed in this thesis, as defined in the SLC guideline, follows the Linear Sequential Model, which is a straightforward software development model, similar to the Waterfall Model. These linear models only allow developers to move onto the next phase as soon as the previous one finishes, and developers cannot go back to previous phases. While they are a suitable and sufficient choice for this project, larger, more complex projects are likely to encounter problems progressively [84]. The authors conclude that the Waterfall Model is not suitable to be used in large-scale software development [84]. Agile software development is an alternative approach that is often implemented in organizations, sometimes even in combination with a linear model [85]. Since this thesis only applies a Linear Sequential Model, it may have limitations in regards to projects using a different methodology, such as an agile one. Additionally, due to the fact that a prototype of the ML classification pipeline [11] existed prior to the establishment of the software development plan, the execution of the plan was more manageable.

Critically, despite efforts to design a user-friendly CDSS, real-world integration into clinical workflows presents significant challenges that require more testing and refinement [9]. During this project, only limited testing was possible. Nonetheless, usability tests were performed during the UE guideline application in order to identify possible use-related risks and to test the user interface. Unfortunately, this project conducted those tests with users not belonging to the intended user group as set by the Use Specification in the UE guideline. Ideally, only physicians should have been the group of people to perform the usability tests. Since that was not the case, those tests did not correctly replicate the intended use environment.

Similarly to "the virtual doctor", the research project dealing with the development of a CDSS [10], this project aims to adapt a similar concept and follow in its footsteps in a way that not only physicians can profit from the provided diagnostic assistance, but also patients at risk for developing skin cancer. It not only aligns with the important goal of advancing medical diagnostics but also serves as a testament to the potential of integrating artificial intelligence in healthcare.

In comparison to studies which primarily focused on the technical aspects of machine learning models for skin lesion classification [7], [11], this work extends the discussion to include the full software development process. While previous research has highlighted the accuracy and efficiency of machine learning algorithms in diagnosing skin cancer [6], the findings here contribute to a broader understanding of how such technologies could be systematically developed and documented to allow integration into the healthcare sector. By emphasizing the importance of a well-documented development process, this thesis also addresses the challenge to achieve regulatory compliance in academic medical software development.

While the software developed in this project has promising results in skin lesion classification, it is important to note that achieving near-regulatory compliance does not equate to immediate clinical deployment. This distinction highlights the ongoing need for further rigorous validation to meet the full spectrum of clinical and regulatory requirements, especially since the strict regulations for medical software development can pose additional significant barriers such as complex approval processes and high compliance costs, which can deter rapid innovation and implementation [9], therefore limiting this study in that perspective. Still, the project’s approach to documenting and adhering to specified guidelines serves as a valuable blueprint for future developments in medical software, suggesting that the path to clinical adoption may be accelerated by adopting such comprehensive development strategies.

Consequently, this project not only complements existing research by providing an exemplary precedent and roadmap for the development of similar applications, but also for future projects aiming to navigate the complex process of medical software certification, documentation, and deployment. The results underscore the potential of structured, guideline-based approaches in accelerating the development of medical software, thereby narrowing the gap to achieving regulatory standards, marking a crucial step towards clinical adoption. By demonstrating a viable pathway to approaching regulatory compliance, this project could assist potential future work in the integration of AI in medical fields and a critical reevaluation of how academic innovations can be translated into healthcare technologies more effectively. This essentially progresses research in a way that could positively contribute to improvements in patient care and outcomes.

Bibliography

- [1] WHO – World Health Organization. *Skin cancer*. URL: <https://www.iarc.who.int/cancer-type/skin-cancer/#:~:text=Introduction&text=Skin%20cancers%20are%20the%20most,people%20died%20from%20the%20disease>. (visited on 04/06/2024).
- [2] *Why Is Melanoma So Dangerous as a Cancer?* 2022. URL: <https://www.usdermatologypartners.com/blog/why-is-melanoma-dangerous/> (visited on 03/14/2024).
- [3] Hensin Tsao, Michael B. Atkins, and Arthur J. Sober. “Management of Cutaneous Melanoma”. In: *The New England Journal of Medicine* 351.10 (2004), pp. 998–1012. DOI: 10.1056/NEJMra041245. URL: <https://doi.org/10.1056/nejmra041245>.
- [4] Qianwei Liu et al. “Mapping the landscape of artificial intelligence in skin cancer research: a bibliometric analysis”. In: *Frontiers in oncology* 13 (2023). 13 Oct. 2023, p. 1222426. DOI: 10.3389/fonc.2023.1222426.
- [5] Sai Nitisha Tadiboina. “The Use Of AI In Advanced Medical Imaging”. In: *Journal of Positive School Psychology* 6.11 (2022), pp. 1939–1946.
- [6] Dr. Kashini Andrew et al. *Improved AI tool shows high sensitivity rates in skin cancer detection*. DOI: <https://doi.org/10.55788/f9840392>. URL: <https://conferences.medicom-publishers.com/specialisation/dermatology/eadv-2023/improved-ai-tool-shows-high-sensitivity-rates-in-skin-cancer-detection/> (visited on 04/06/2024).
- [7] Bhuvaneshwari Shetty et al. “Skin lesion classification of dermoscopic images using machine learning and convolutional neural network”. In: *Scientific Reports* 12.1 (Oct. 2022). DOI: 10.1038/s41598-022-22644-9. URL: <https://www.nature.com/articles/s41598-022-22644-9.pdf>.
- [8] Sabrina Celine Holst. “Software Life Cycle Guideline for Software as a Medical Device - Implementing a Prototype App”. MA thesis. Philipps University Marburg, 2021.
- [9] Mona Riemenschneider, Joachim Wienbeck, André Scherag, et al. “Data Science for Molecular Diagnostics Applications: From Academia to Clinic to Industry”. In: *Systems Medicine* 1.1 (2018), pp. 13–17. DOI: 10.1089/sysm.2018.0002. eprint: <https://doi.org/10.1089/sysm.2018.0002>. URL: <https://doi.org/10.1089/sysm.2018.0002>.
- [10] Sebastian Spänig, Agnes Emberger-Klein, Jan-Peter Sowa, et al. “The virtual doctor: An interactive clinical-decision-support system based on deep learning for non-invasive prediction of diabetes”. In: *Artificial Intelligence in Medicine* 100 (2019), p. 101706. ISSN: 0933-3657. DOI: <https://doi.org/10.1016/j.artmed.2019.101706>. URL: <https://www.sciencedirect.com/science/article/pii/S0933365719301083>.

-
- [11] Dmitry Degtyar. “Machine Learning based skin cancer screening”. Philipps University Marburg, 2023.
- [12] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. “The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions”. In: *Scientific Data* 5 (2018), p. 180161. DOI: 10.1038/sdata.2018.161. URL: <https://doi.org/10.1038/sdata.2018.161>.
- [13] *Hautkrebs Fakten & Statistiken*. 2023. URL: <https://www.skincancer.org/de/skin-cancer-information/skin-cancer-facts> (visited on 01/14/2024).
- [14] Statistisches Bundesamt (Destatis). “Zahl der stationären Hautkrebsbehandlungen binnen 20 Jahren um 75 % gestiegen”. In: *Statistisches Bundesamt* (2023). URL: https://www.destatis.de/DE/Presse/Pressemitteilungen/Zahl-der-Woche/2023/PD23_21_p002.html (visited on 03/18/2024).
- [15] K. M. Hosny, M. A. Kassem, and M. M. Fouad. “Classification of Skin Lesions into Seven Classes Using Transfer Learning with AlexNet”. In: *Journal of digital imaging* (2020). URL: <https://doi.org/10.1007/s10278-020-00371-9>.
- [16] Leonie Pape-Werlich, Dres. Schlegel, and Schmidt Medizinische Kommunikation GmbH. *Warnsignal der Haut - aktinische Keratose*. 2024. URL: <https://www.tk.de/techniker/gesundheit-und-medizin/behandlungen-und-medizin/haut-und-geschlechtskrankheiten/warnsignal-der-haut-aktinische-keratose-2017604> (visited on 03/10/2024).
- [17] Dr. med. Andreas Arnold. “Diagnostik und Therapie von nicht melanozytären Hauttumoren Basalzellkarzinom und Plattenepithelkarzinom im Fokus”. In: *hautnah dermatologie* (2016). URL: <https://doi.org/10.1007/s15012-016-2070-6>.
- [18] Lowell A. Goldsmith, Stephen I. Katz, Barbara A. Gilchrest, et al. In: *Fitzpatrick’s Dermatology in General Medicine, 8e*. New York, NY: The McGraw-Hill Companies, 2012. URL: accessmedicine.mhmedical.com/content.aspx?aid=1000185991.
- [19] Denise M. Aaron. *Seborrhoische Keratosen*. 2022. URL: <https://www.msmanuals.com/de-de/profi/erkrankungen-der-haut/gutartige-hauttumoren,-wucherungen-und-vaskul%C3%A4re-l%C3%A4sionen/seborrhoische-keratosen> (visited on 03/10/2024).
- [20] David J Myers and Emily P Fillman. “Dermatofibroma”. In: *StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing* (2022). URL: <https://www.ncbi.nlm.nih.gov/books/NBK470538/>.
- [21] Prof. Dr. med. Peter Altmeyer, Prof. Dr. med. Martina Bacharach-Buhles, and Alexandros Zarotis. *Dermatofibrom*. 2020. URL: <https://www.altmeyers.org/de/dermatologie/dermatofibrom-1752> (visited on 03/10/2024).
- [22] *Melanoma Overview: A Dangerous Skin Cancer*. 2022. URL: <https://www.skincancer.org/skin-cancer-information/melanoma/> (visited on 03/11/2024).

-
- [23] *What is melanoma skin cancer?* 2024. URL: <https://www.cancerresearchuk.org/about-cancer/melanoma/about> (visited on 03/11/2024).
- [24] A. Hunter Shain and Boris C. Bastian. “From melanocytes to melanomas”. In: *Nature Reviews Cancer* 16.6 (Apr. 2016), pp. 345–358 (). DOI: 10.1038/nrc.2016.37.
- [25] *Neun Zahlen zum Thema Hautkrebs*. 2021. URL: <https://www.aok.de/pk/magazin/koerper-psyche/krebs/neun-zahlen-zum-thema-hautkrebs/> (visited on 03/18/2024).
- [26] *Vascular Lesions*. URL: <https://www.mayoralderm.com/vascular-lesions/> (visited on 03/11/2024).
- [27] Abhishek et al. Bhattacharya. “Precision Diagnosis Of Melanoma And Other Skin Lesions From Digital Images”. In: *AMIA Joint Summits on Translational Science proceedings* 2017 (July 2017), pp. 220–226.
- [28] *Tests for skin cancer*. 2023. URL: <https://www.cancerresearchuk.org/about-cancer/skin-cancer/getting-diagnosed/tests> (visited on 03/10/2024).
- [29] Giuseppe Argenziano and et al. “Dermoscopy of pigmented skin lesions: Results of a consensus meeting via the Internet”. In: *Journal of The American Academy of Dermatology* 48.5 (2003), pp. 679–693. DOI: 10.1067/mjd.2003.281.
- [30] Institute of Medicine (US) Committee on Medicare Coverage Extensions. *Extending Medicare Coverage for Preventive and Other Services*. Screening for Skin Cancer. Ed. by M.J. Field, R.L. Lawrence, and L. Zwanziger. National Academies Press (US), 2000. Chap. 3. URL: <https://www.ncbi.nlm.nih.gov/books/NBK225258/>.
- [31] Matt Crabtree. *What is Machine Learning? Definition, Types, Tools & More*. 2023. URL: <https://www.datacamp.com/blog/what-is-machine-learning> (visited on 03/07/2024).
- [32] Crypto1. *Gradient Descent Algorithm: How does it Work in Machine Learning?* 2024. URL: <https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/> (visited on 03/07/2024).
- [33] Johanna Ronsdorf. *Microsoft erklärt: Was ist Deep Learning? Definition & Funktionen von DL*. 2020. URL: <https://news.microsoft.com/de-de/microsoft-erklaert-was-ist-deep-learning-definition-funktionen-von-dl/> (visited on 03/13/2024).
- [34] *Was ist ein Convolutional Neural Network?* URL: <https://de.mathworks.com/discovery/convolutional-neural-network.html> (visited on 03/11/2024).
- [35] Zoumana Keita. *An Introduction to Convolutional Neural Networks (CNNs)*. 2023. URL: <https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns> (visited on 03/11/2024).

-
- [36] R. Yamashita, M. Nishio, R.K.G. Do, et al. “Convolutional neural networks: an overview and application in radiology”. In: *Insights into Imaging* 9 (2018), pp. 611–629. DOI: 10.1007/s13244-018-0639-9. URL: <https://doi.org/10.1007/s13244-018-0639-9>.
 - [37] Rany ElHousieny. “Understanding Neural Networks and Deep Learning”. In: *Medium.com – Techiepedia* (2024). URL: <https://levelup.gitconnected.com/understanding-neural-networks-and-deep-learning-0039ab6bf2a0>.
 - [38] Sneha H.L. “2D Convolution in Image Processing”. In: *EETech Media, LLC*. (2018). URL: <https://www.allaboutcircuits.com/technical-articles/two-dimensional-convolution-in-image-processing/>.
 - [39] Piotr Skalski. “Gentle Dive into Math Behind Convolutional Neural Networks”. In: *Medium.com – Towards Data Science* (2019). URL: <https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9>.
 - [40] Lei Qu, Changfeng Wu, and Liang Zou. “3D Dense Separated Convolution Module for Volumetric Medical Image Analysis”. In: *Applied Sciences* 10.2 (2020). ISSN: 2076-3417. DOI: 10.3390/app10020485. URL: <https://www.mdpi.com/2076-3417/10/2/485>.
 - [41] Shiv Ram Dubey, S. K. Singh, and Bidyut Baran Chaudhuri. “Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark”. In: *Cornell University* (Sept. 2021). DOI: 10.48550/arxiv.2109.14545. URL: <http://arxiv.org/pdf/2109.14545>.
 - [42] Sai Balaji. *Binary Image classifier CNN using TensorFlow*. 2020. URL: <https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697> (visited on 03/08/2024).
 - [43] M. Li, Y. Jiang, Y. Zhang, et al. “Medical image analysis using deep learning algorithms”. In: *Frontiers in Public Health* 11:1273253 (2023). DOI: 10.3389/fpubh.2023.1273253. URL: <https://doi.org/10.3389/fpubh.2023.1273253>.
 - [44] *Medical devices*. URL: <https://www.ema.europa.eu/en/human-regulatory-overview/medical-devices> (visited on 03/27/2024).
 - [45] Daniel Reinsch. *Software als Medizinprodukt – Software as Medical Device*. 2023. URL: <https://www.johner-institut.de/blog/regulatory-affairs/software-als-medizinprodukt-definition/> (visited on 03/24/2024).
 - [46] Anne-Sophie Grell. *SaMD versus MDSW: what’s the difference between Software as a Medical Device and Medical Device SoftWare?* 2021. URL: <https://qbdgroup.com/en/blog/samd-mdsw-difference/> (visited on 03/24/2024).
 - [47] Wade Schroeder. *Ultimate Guide to Software as a Medical Device (SaMD)*. 2023. URL: <https://www.greenlight.guru/blog/samd-software-as-a-medical-device> (visited on 03/25/2024).

-
- [48] *Is Ada a medical device?* URL: <https://ada.com/help/is-ada-a-medical-device/> (visited on 03/31/2024).
- [49] Luca Salvatore. *Medical Device Regulation MDR – Medizinprodukteverordnung (2017/745)*. 2024. URL: <https://www.johner-institut.de/blog/regulatory-affairs/medical-device-regulation-mdr-medizinprodukteverordnung/> (visited on 03/09/2024).
- [50] Andy Kahles et al. “Struktur und Inhalt der EU-IVDR: Bestandsaufnahme und Implikationen für die Pathologie”. German. In: *Pathologie (Heidelberg, Germany)* 43.5 (2022), pp. 351–364. DOI: 10.1007/s00292-022-01077-1. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9118816/>.
- [51] *In vitro diagnostics*. URL: https://www.who.int/health-topics/in-vitro-diagnostics#tab=tab_1,%20https://health.ec.europa.eu/medical-devices-sector/overview_en (visited on 03/11/2024).
- [52] Dr. Kai Moritz Eder. *In vitro Diagnostic Medical Device Regulation (IVDR) – Verordnung über In-vitro-Diagnostika (2017/746)*. 2024. URL: <https://www.johner-institut.de/blog/regulatory-affairs/ivdr-in-vitro-diagnostic-device-regulation/> (visited on 03/11/2024).
- [53] *ISO 9001:2015 – Quality management systems – Requirements*. Standard. Geneva, Switzerland: International Organization for Standardization, 2015.
- [54] *ISO 13485:2016 – Medical devices – Quality management systems – Requirements for regulatory purposes*. Standard. Geneva, Switzerland: International Organization for Standardization, 2016.
- [55] *IEC 62304:2006+A1:2015 – Medical device software - Software life-cycle processes*. Tech. rep. International Electrotechnical Commission, 2015.
- [56] *IEC 82304-1:2016 – International Standard Health software*. Tech. rep. International Electrotechnical Commission, 2016.
- [57] Richard Bellairs. *What Is IEC 62304? Overview + Compliance Tips*. 2019. URL: <https://www.perforce.com/blog/qac/what-iec-62304> (visited on 03/12/2024).
- [58] Daniel Reinsch. *IEC 82304 – Was die Norm zu „Health Software“ fordert*. 2016. URL: <https://www.johner-institut.de/blog/iec-62304-medizinische-software/iec-82304/> (visited on 03/12/2024).
- [59] *ISO/IEC 25010:2011 – Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*. Tech. rep. The British Standards Institution, 2011.
- [60] Jill Britton. *What Is ISO 25010?* 2021. URL: <https://www.perforce.com/blog/qac/what-is-iso-25010> (visited on 03/10/2023).
- [61] Sascha Block. *ISO 25010 – Kriterien zur Qualitaet von Software*. 2023. URL: <https://inztitut.de/blog/glossar/iso-25010> (visited on 03/10/2024).

-
- [62] *ISO 14971:2019 – Medical devices – Application of risk management to medical devices*. Standard. Geneva, Switzerland: International Organization for Standardization, 2019.
 - [63] *CEN ISO/TR 24971:2020 – Medical devices – Guidance on the application of ISO 14971*. Technical Report. Geneva, Switzerland: International Organization for Standardization, 2020.
 - [64] *ISO 14971 Explained*. URL: <https://standard.com/en-sg/blog/iso-14971/> (visited on 03/11/2024).
 - [65] *IEC 62366-1 (2015) – Medical devices - Part 1: Application of usability engineering to medical devices (Edition 1.0)*. Tech. rep. International Electrotechnical Commission, 2015.
 - [66] *IEC TR 62366-2 (2016) – Medical devices - Part 2: Guidance on the application of usability engineering to medical devices (Edition 1.0)*. Tech. rep. International Electrotechnical Commission, 2016.
 - [67] *Usability & IEC 62366-1*. URL: <https://www.johner-institut.de/blog/category/iec-62366-usability/> (visited on 03/12/2024).
 - [68] Anne-Christin Hauschild, Lisa Eick, Joachim Wienbeck, et al. “Fostering reproducibility, reusability, and technology transfer in health informatics”. In: *iScience* 24.7 (July 2021), p. 102803. DOI: 10.1016/j.isci.2021.102803.
 - [69] Anne-Christin Hauschild, Roman Martin, Sabrina Celine Holst, et al. “Guideline for software life cycle in health informatics”. In: *iScience* 25.11 (Nov. 2022), p. 105534. DOI: 10.1016/j.isci.2022.105534.
 - [70] Roman Martin, Anne-Christin Hauschild, Robin Gottschalk, et al. “Guideline for Risk Managament Manuscript”. In: *iScience* ().
 - [71] Dr. Dominik Heider, Dr. Anne-Christin Hauschild, Dr. Joachim Wienbeck, et al. “Deliverable Result D3.8 – Manuscript on Usability Process.” In: *FeatureCloud* (2023). URL: <https://featurecloud.eu/results/> (visited on 04/10/2024).
 - [72] Vanessa Klemmt. “Usability Engineering Guideline for Software as a Medical Device – Implementing an Interactive xAI Platform supporting Medical Decision-Making”. MA thesis. Philipps University Marburg, 2021.
 - [73] Leodanis Pozo Ramos. *Qt Designer and Python: Build Your GUI Applications Faster*. URL: <https://realpython.com/qt-designer-python/> (visited on 03/17/2024).
 - [74] Nisith Kumar Pati et al. “Oversampled Two-dimensional Deep Learning Model for Septenary Classification of Skin Lesion Disease”. In: *National Academy Science Letters* (2022). DOI: 10.1007/s40009-022-01175-x. URL: <https://doi.org/10.1007/s40009-022-01175-x>.

-
- [75] Khairul Islam et al. “Melanoma Skin Lesions Classification using Deep Convolutional Neural Network with Transfer Learning”. In: *IEEE* (Apr. 2021). DOI: 10.1109/caida51941.2021.9425117. URL: <https://doi.org/10.1109/caida51941.2021.9425117>.
- [76] Peshawa Muhammad Ali and Rezhna Faraj. *Data Normalization and Standardization: A Technical Report*. Tech. rep. Jan. 2014. DOI: 10.13140/RG.2.2.28948.04489.
- [77] Ziad Alqadi et al. “Features Analysis of RGB Color Image based on Wavelet Packet Information”. In: *International Journal of Computer Science and Mobile Computing* (3 Mar. 2020), pp. 149–159.
- [78] Jason Brownlee. *A Gentle Introduction to Transfer Learning for Deep Learning*. 2019. URL: <https://machinelearningmastery.com/transfer-learning-for-deep-learning/> (visited on 03/15/2024).
- [79] Hee Jin Kim et al. “Transfer learning for medical image classification: a literature review”. In: *BMC Medical Imaging* 22.1 (Apr. 2022). DOI: 10.1186/s12880-022-00793-7. URL: <https://doi.org/10.1186/s12880-022-00793-7>.
- [80] Hee Jin Kim et al. “Transfer learning for medical image classification: a literature review”. In: *BMC Medical Imaging* 22.1 (Apr. 2022). DOI: 10.1186/s12880-022-00793-7. URL: <https://doi.org/10.1186/s12880-022-00793-7>.
- [81] Kensuke Nakamura et al. “Learning-Rate Annealing Methods for Deep Neural Networks”. In: *Electronics* 10.16 (Aug. 2021), p. 2029. DOI: 10.3390/electronics10162029. URL: <https://www.mdpi.com/2079-9292/10/16/2029/pdf>.
- [82] Adeola Adenubi, Ayorinde Oduroye, and Adeniyi Akanni. “ARTIFICIAL INTELLIGENCE (AI) IN HEALTHCARE: TRANSFORMING DIAGNOSIS AND TREATMENT”. In: *ResearchGate* (Mar. 2024).
- [83] Robin Gottschalk. “Risk Management Guideline for Software as a Medical Device and Example Implementation for a Federated Learning Application”. MA thesis. Philipps University Marburg, 2021.
- [84] Kai Petersen, Claes Wohlin, and Dejan Baca. “The Waterfall Model in Large-Scale Development”. In: *Product-Focused Software Process Improvement*. Ed. by Frank Bomarius, Markku Oivo, Päivi Jaring, et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 386–400. ISBN: 978-3-642-02152-7.
- [85] Alok Mishra and Yehia Ibrahim Alzoubi. “Structured software development versus agile software development: a comparative analysis”. In: *International Journal of System Assurance Engineering and Management* 14.4 (Aug. 2023), pp. 1504–1522. ISSN: 0976-4348. DOI: 10.1007/s13198-023-01958-5. URL: <https://doi.org/10.1007/s13198-023-01958-5>.

A. Appendix

The attachments included below are extracts of the wiki and constitute an important reference material for this thesis. The wiki can be accessed only via an invite. If invited, it is found under the following link:

<https://scdsoftware.atlassian.net/wiki/spaces/SD/overview>

The source code for the software is stored in this HHU Git-repository, also only accessible via invite:

<https://git.hhu.de/buy09tay/software-for-skin-cancer-detection>

To request an invitation to either one of the two platforms, please contact the following e-mail address:

Mohammad.Sharabati@uni-duesseldorf.de

A.1. Quality Management System Project Execution Extract

Software for Skin Cancer Detection – Software for Skin Cancer Detection

#	Item	Comment
2	Description of the development environment	<ul style="list-style-type: none">• Name: PyCharm• Code editor:<ul style="list-style-type: none">• Smart code completion• Code analysis and inspections• Code navigation and search• Code refactoring tools• Built-in version control:<ul style="list-style-type: none">• Integration with version control systems like Git• Visual diff and merge tools• Debugger:<ul style="list-style-type: none">• Advanced debugging features for Python applications• Visual debugging tools• Virtual environments:<ul style="list-style-type: none">• Support for managing Python virtual environments• Integration with popular package managers like pip• Database tools:<ul style="list-style-type: none">• Tools for working with databases, including SQL code editing and data manipulation• Support for various database systems• Test tools:<ul style="list-style-type: none">• Integration with testing frameworks (e.g., unittest, pytest)• Tools for running and debugging tests• Frameworks and libraries:<ul style="list-style-type: none">• Support for various Python frameworks and libraries• Integration with technologies like NumPy, pandas, and others• Scientific tools:<ul style="list-style-type: none">• Integration with scientific tools and libraries for data science and analysis• UI/UX:<ul style="list-style-type: none">• Quick access to frequently used actions and tools

Quality Management System Guideline – 9

#	Item	Comment
		<ul style="list-style-type: none"> • Plugins: <ul style="list-style-type: none"> • A variety of plugins to extend functionality for different languages and technologies • Code quality and analysis: <ul style="list-style-type: none"> • Integration with tools for code quality analysis • Code metrics and inspection reports • Project management: <ul style="list-style-type: none"> • Integration with issue trackers and project management tools • Support for collaboration with team members • Documentation: <ul style="list-style-type: none"> • Tools for generating and viewing code documentation • Cloud integration: <ul style="list-style-type: none"> • Integration with cloud platforms for deployment and monitoring <p>Source: PyCharm documentation, JetBrains WebPage</p>
3	Description of build and integration steps	<ol style="list-style-type: none"> 1. Pull software code from git repository: https://git.hhu.de/buy09tay/software-for-skin-cancer-detection 2. View "README" file and read Installation-section 3. Utilize pip for package importing <p>Version 0.1.0</p>
4	Description of requirements handling	<ul style="list-style-type: none"> • Requirements were set in regards to user skill, competence, and stakeholder wishes • High-level requirements were prioritized, followed by risk-mitigating technical requirements. Visual and aesthetic requirements were prioritized last. • Requirements are documented in Requirements Specification
5	Plan for testing	Only punctual, functional tests are necessary in regards to software purpose.
6	Handling of changes	Changes during the development will be documented in each version.

A.2. Risk Policy

Software for Skin Cancer Detection – Software for Skin Cancer Detection

Risk Policy

Rationale

The three-class severity and probability classification proposed in the guideline is used for risk policy. The severity classes are Negligible (no damage/very little damage), Moderate (reversible injury occurs) and Significant (death or irreversible injury), and the probability of the occurrence is categorized as either High (occurs frequently), Medium (occurs a few times during the lifetime of the SaMD) or Low (occurs very rarely).

The resulting Risk matrix can be found below. **Dark-grayed out cells are unacceptable risks, while white cells are acceptable.** As the software is only a supportive instrument, it is very unlikely that risks exist that have a high frequency and are significant or moderate, or medium frequent risks with a moderate/significant severity.

Risk matrix

	Negligible	Moderate	Significant
High			
Medium			
Low			

Severity classes

Severity Level of Harm	Description
Negligible	Results in inconvenience or temporary discomfort, no injury
Moderate	Results in injury or impairment requiring or not professional medical care
Significant	Results in permanent impairment or life-threatening injury

View [Use related Risk Analysis](#).

A.3. Risk Table

Final identified risks, 17.02.2024

The table below is exemplary and not exhaustive. It serves as a sample risk table.

Risk ID	Component	Error (Cause)	Hazard	Damage
R1	Database	Database not reachable or takes too long to answer	H1	No (or delayed) treatment
R2	Application	Incomplete form submission (missing inputs)	H1	No (or delayed) treatment
R3	Database Security	Malicious SQL injection	H1, H2, H3, H4	No treatment, false treatment, system manipulation or user data violation
R4	Database Security	Unauthorized access to stored patient diagnoses	H4	Privacy violation
R5	Database Security	Malicious user floods system with flake data	H3	System damage
R6	Application	User provides wrong name	H1, H2	No or false treatment
R7	Application	User uploads garbage data (e.g. unrelated images)	H1, H2	No or false treatment
R8	Communication	User provides wrong contact info	H1, H2, H4	No treatment, false treatment or privacy violation

A.4. Jira Project Plan

Jira Software

- Your work ▾ Projects ▾ Dashboards ▾ Teams ▾ Plans ▾ Apps ▾
- Create

Q Search

No users left

⚙️ ? 🔔 ⭐ 🔄 ...

View settings

Projects / Software for Skin Cancer Detection

PROJ Board

MS Add people

Q

GROUP BY	None ▾	Insights	View settings
BIS ZUM NÄCHSTEN MEETING	DONE ✓	Q See all Done issues	

GENERELLE AUFGABEN 5	LEITFADEN-AUFGABEN 5	IN PROGRESS 2
Dokumentation Vorlage finden <input checked="" type="checkbox"/> S4SCD-9	PE-1 Softwaredevelopment Plan erstellen <input checked="" type="checkbox"/> S4SCD-4	PP-3 List of needed Documents erstellen <input checked="" type="checkbox"/> S4SCD-3
Anfangen zu programmieren <input checked="" type="checkbox"/> S4SCD-11	SLC-2 Software Development Plan beschreiben <input checked="" type="checkbox"/> S4SCD-6	Die Präsentation bis zum 30.11.! <input checked="" type="checkbox"/> S4SCD-19
Auf Git Unterordner und Organisation <input checked="" type="checkbox"/> S4SCD-13	SLC-3 Software Requirements Analysis definieren <input checked="" type="checkbox"/> S4SCD-7	
QMS Checkliste erstellen <input checked="" type="checkbox"/> S4SCD-15	SLC-4 Software Design <input checked="" type="checkbox"/> S4SCD-8	
Checklisten zusammenführen		

A.5. Confluence Overview

Confluence

Home

Recent

Spaces

Teams

Apps

Templates

Create

Upgrade

Search

Unstar this space

Share

...

Software for Skin Cancer Detection

All content

Automation

Calendars

Space settings

SHORTCUTS

Task-Board

CONTENT

Quality Management System Guideline

Software Life Cycle Guideline

Risk Management Guideline

Usability Engineering Process Guideline

List of abbreviations

Bibliography

Invite people

Software for Skin Cancer Detection

This Wiki-Page will serve as the homepage for this Wiki. In here, there will be links to each guideline and other relevant documentation for this thesis.

QMS Guideline

SLC Guideline

RM Guideline

UE Guideline

List of abbreviations

Bibliography

XIV

A.6. Risk Control Measures

Similar to risk control in the risk management process of ISO 14971, the usability norm requires to implement at least one of the risk control measures, which are listed according to their priority in the table below [7, table 10]:

Risk Control Measure Type	Description	Examples
Inherent safety by design	User interface is designed in a way that prevents use errors	<ul style="list-style-type: none">• System does not allow incorrect input or selection• Improve detectability or readability of controls• Remove features that can be mistakenly selected
Protective measure	Built-in protections against use errors as part of the user interface	<ul style="list-style-type: none">• Dialogue that requires the user to actively confirm a critical action in order to proceed
Information for safety	Warnings or instructions on correct use that inform the user on potential risks	<ul style="list-style-type: none">• Training sessions• Online user manual• Highlight problematic inputs / results without requiring any confirmation

If possible, *Inherent safety by design* should be the preferred measure because a redesign of the user interface will most likely reduce or remove the risk [7]. However, this might only sometimes be possible. Therefore, the second option is protective measures. In our case for software, there are no protective measures in the actual sense, such as protective insulation. Instead, *confirmation prompts* can be implemented for critical actions that the user must actively confirm to proceed.

A.7. Document Management

Software for Skin Cancer Detection – Software for Skin Cancer Detection

Document Management

Procedure DM-1: Assignment

This procedure identifies one person responsible for the documentation. The document representative is Mohammad Sharabati.

Procedure DM-2a: Documentation platform

The chosen platform for documentation purposes is Confluence Wiki. Essential requirements are met, those include:

- Versioning, including versioning of renamed and deleted documents (no data loss).
- Editing and document access only by authorized users.

Procedure DM-2b: Documentation platform (Code)

The chosen platform for code management is Git/GitLab (git.hhu.de), provided by the Heinrich-Heine University in Düsseldorf. Essential requirements are met, including:

- Code management properties.
- Versioning.
- Possibility of storing intermediate states in a referenceable way.
- Prevention of code loss and simultaneous work by more than one developer on the same code.

Procedure DM-3: Documentation life cycle

The monitoring of life cycle documents is performed by the documentation representative mentioned above. All documents are created by the documentation representative and there is no special procedure; if a document must be created, the responsible person creates it and documents its existence. Additionally, documents have to be versioned and older versions do not need to be accessible to people not assigned to the project.

Backups are performed by Confluence and do not require any interference, guaranteeing consistent storage.

Procedure DM-4: Documentation change

If any changes must be made, refer to procedure DM-3 and [Configuration and Change Management](#).

Procedure DM-5: Meeting minutes

Meetings are planned by the person who requested it unless discussed otherwise. Documentation is not necessary unless all parties agree that documentation is required.

Procedure DM-6: Standard operating procedure documentation

SOPs are documented and inserted here.

#	Headline	Content
1	Title	Standard operating procedure: Weekly Update Meeting
2	Department, date, ID	Lehrstuhl Prof. Dr. Heider, 18.01.2024, SOP-01
3	Purpose	Update supervisors & stakeholders with thesis progress in a virtual or live meeting.
4	Scope	This procedure applies to all meetings.
5	Definitions	ID - Identification
6	Procedure	Firstly, settle on a weekly appointment time. Then meet with the participants and update them. The content of the meeting does not have to be recorded and the weekly appointment time may be changed if all participants agree permanently or occasionally.

A.8. Quality Management System Project Planning Extract

Software for Skin Cancer Detection – Software for Skin Cancer Detection

Procedure PP-2: Prepare document storage

A folder with the name “Software for Skin Cancer Detection” was created in this Confluence Wiki and in the documentation platforms described in [Documentation Management](#) under Procedure DM-2. Further documents and records will be saved in subfolders accordingly.

Procedure PP-3: List of documents

A list of necessary documents that will be created during the execution of this project is listed below in an effort to make sure that no document will be forgotten. All additions, removals and changes made are documented in each version and in archives. The initial list is created in the very first version of this Wiki-page.

List of documents:

- [Safety Class Determination](#)
- [Intended Purpose](#)
- [Software Development Plan](#), consisting of:
 - [Process Description Plan](#)
 - [Development Plan](#)
- [System and Software Requirements Specification](#)
- [Software Architecture Description](#)
- [System and Software Design](#)
- [Mockup and Class Diagram](#)
- [Implementation, Testing, and Verification Document](#)
- [Software problem-solving process](#)
- [Risk Preparation: Planning Document](#)
- [Risk Policy](#)
- [Risk Monitoring](#)
- [Use specification](#)
- [Summative Evaluation](#)
- [List of abbreviations](#)

A.9. Intended Purpose

Software for Skin Cancer Detection – Software for Skin Cancer Detection

Intended Purpose / Use Main Page

The app's purpose is to support medical professionals with the diagnosis of skin cancer in patients. By connecting a dermatoscope and pointing it at a spot on the skin, the app can calculate a risk score of the most prominent skin lesion types in a matter of seconds. This would speed up the process of detecting skin cancer immensely. However, for a final valid diagnosis, more tests need to be conducted and physicians should not neglect their duties and still take the standard clinical measures. The app should only provide users with supplementary information and therefore is in no way a substitute for a clinical examination.

A.10. Safety Class Determination

Software for Skin Cancer Detection – Software for Skin Cancer Detection

Safety Class Determination

MDR qualification and classification process

As this software has a medical purpose and does not only manage data but also assists with a diagnosis, a classification for this software is necessary. The process is visualized below:

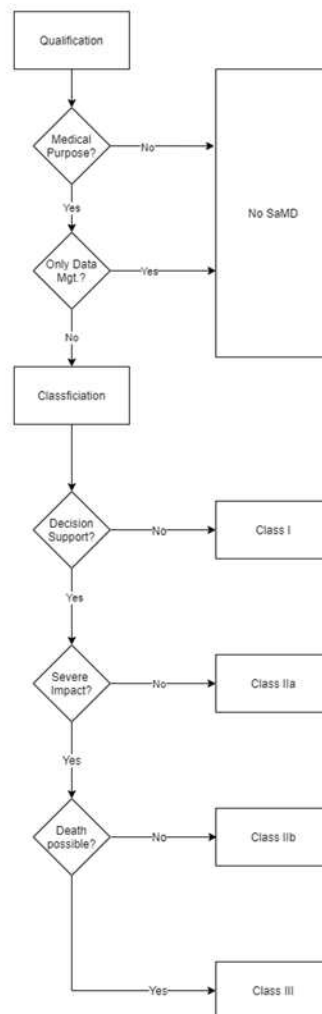


Fig. 1: Safety Class Determination process. [6]

As stated in the [Use specification page](#), this software will assist physicians and support their decisions in diagnosing skin cancer in patients. Additionally, a misdiagnosis as a result of an incorrect software output

might have severe impacts for the patients well-being and health, which is why a classification of 2b would have to be achieved.

However, as this software is only a supportive instrument in a full diagnosis, which has to be conducted by a medical professional, such health consequences would not account to the software use alone, resulting in placing this software's safety class into the 2a category.

A.11. Software Development Plan

Software for Skin Cancer Detection – Software for Skin Cancer Detection

Development Plan

The [Process Description Plan](#) lays a foundation for the finalization of the Software Development Plan below.

This detailed plan represents all relevant development phases and their tasks, including the estimated duration it takes to finish a task and what has to be finished before another task can begin.

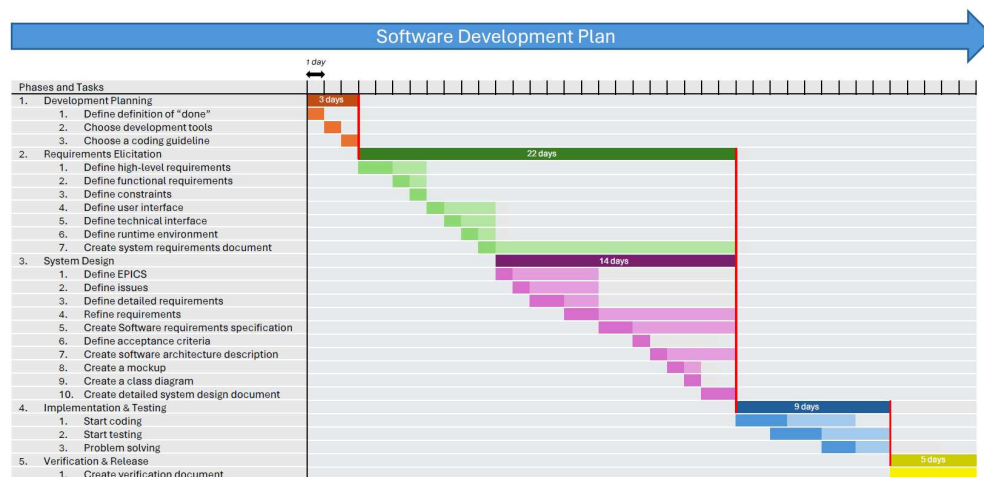


Fig. 1: Software Development Plan. Phases and their task names are on the left, with their respective duration in the Gantt-chart on the right. Some tasks have lighter colored bars, indicating the maximum additional days that the estimated duration of those tasks may exceed. Red lines symbolize the fact that all phases (except 2. and 3.) may only commence when the phase at the top of the line is finished in its entirety. Some tasks are also dependent on a previously finished task (no red lines for overview aesthetic reasons).

The Software Development Plan consists of five phases and is based on the Linear Sequential Model, meaning that a phase can only begin if the previous phase has been finished successfully.

In this plan, task seven of phase two is the only task exempt from the above mentioned rule, as this task does not have a big impact on the following phase since it only has a documentation and summarization purpose, just like task ten in phase three. However, all tasks, up until task ten, phase three, must be finalized before the next phase starts, as this phase relies on those results.

The **Definition of 'done'** (DoD) is used to check if a requirement or its related issues are fully implemented and can be tagged as completed within Confluence and Jira [5]. Within this project, a requirement is fully implemented, sufficing the DoD, if:

- The code is coding guideline compliant.
- The code is consistent, unambiguous, and clearly identifiable.
- A code review was conducted.
- All acceptance criteria are fulfilled.
- A build has been made and deployed on a testing environment.
- All necessary unit, integration, and system tests are passed and documented.

- Traceability to the requirements specification, epics, architecture, tests, and/or the detailed design document is implemented.
- The implementation does not contradict the architecture description or detailed design document.
- The documentation is complete.

Defining a **coding guideline** or convention, including code style, nomenclature, and naming in the development plan, to increase software quality, is recommended in the Software Development Plan [5].

This project defines the following **coding guideline**:

1. Naming Conventions
 - a. US English spelling.
 - b. Usage of clear and descriptive variable and function names that convey their purpose.
 - c. Stay consistent with name conventions throughout the codebase with camelCase for classes, typedefs, etc.
 - d. Avoidance of overly cryptic and abbreviated names, except for one-character names in for-clauses for example.
2. Documentation
 - a. Comments for functions and complex codes in order to explain their purpose and how they work.
 - b. As simple and as short as possible.
 - c. Avoid useless documentation.
3. Avoiding of exceptions and faults
 - a. Exceptions not used for normal control flow, only in exceptional cases.
 - b. Implementation of error handling for unexpected situations or inputs.
4. Ensuring a good coding style and efficient runtime
 - a. Code is indented consistently, tab size is always 4 spaces.
 - b. Long code lines are broken down into multiple lines for readability purposes.
 - c. Single Responsibility Principle: One functions has a single task/responsibility.

This finished document, together with the process description plan, solves tasks 1-3 of phase 1, allowing phase 2 to start.

A.12. Use Specification

Software for Skin Cancer Detection – Software for Skin Cancer Detection

Use specification

Please read [Intended Use](#) first.

Intended medical indication

Screening, prediction and diagnosis of seven types of skin lesions.

Patient population

The patient population includes patients of all age groups, genders, weight ranges, health conditions and disabilities.

However, patients with a darker skin tone and patients with rare skin conditions (Vitiligo, or similar) *may* experience higher misclassification rates.

Parts of the human body or type of tissue applied to

The software, in combination with the dermatoscope, is intended for use on the human skin on any part of the body. For more accurate results, the skin should be hairless (or shaved). In rare cases, the skin must be adequately prepared for use, for example in individuals with a skin condition resulting in excessive calluses production like hyperkeratosis.

User profile

Since user characteristics are highly likely to influence the efficiency and effectiveness as well as the safety of the SaMD, user profiles have to be characterized. The intended user group for this product consists of English speaking physicians, or other similarly skilled individuals with equivalent expertise and educational level.

The software is allowed to be used by other user groups if supervision by a member of the specified user group above is given.

Use environment

This SaMD is intended for use in clinical settings such as hospitals or doctors offices. Aspects like noise levels, lighting and temperature are irrelevant if access to a dermatoscope with a polarization filter is given. A single person suffices in utilization of the software.

Operating principle

Inputs into the software are solely images taken by the dermatoscope, alongside patient information.

The software outputs a numerical risk assessment of seven skin lesion types, with each number representing the risk score of the corresponding skin lesion type. The data can be saved locally and in a database.

A.13. Risk Management Preparation and Planning

Software for Skin Cancer Detection – Software for Skin Cancer Detection

Preparation: Planning Document

Location of RM documents

Documents are stored in this Confluence Wiki under the RM-Guideline section.

Schedule

Phase	Date	Persons involved
Preparation	02.02.2024	Mohammad Sharabati
Analysis	17.02.2024	Mohammad Sharabati
Evaluation	17.02.2024	Mohammad Sharabati
Control	17.02.2024	Mohammad Sharabati
Initial Monitoring	18.02.2024	Mohammad Sharabati
Monitoring Intervals	weekly	Mohammad Sharabati

Intended use table

Project	Documentation-based Machine Learning Software for Skin Cancer Detection - Bachelor's thesis
Preliminary name	Software for Skin Cancer Detection
Medical problem/indication	Supportive diagnosis in classifying skin lesions
Users	Physicians, researchers
Use environment	Doctor's offices, hospitals
Patients	Patients with a medical indication

Risk Management Guideline – 46

A.14. System and Software Requirements Specification

Software for Skin Cancer Detection – Software for Skin Cancer Detection

System and Software Requirements Specification Document

This document consists of all task outputs for phase 2: *Requirements Elicitation* of the [Software Development Plan](#). Since this software is standalone, it is not possible to clearly distinguish between system and software requirements [2]; as such, they are combined.

High-level functional requirements

ID	Requirement	User Story / System Requirement	Priority
1	High-level requirement	As a user, I want to get the different values for the skin lesion classifications.	Necessary
2	High-level requirement	As a user, I want to know what the risk scores indicate and what consequences arise.	Necessary
3	High-level requirement	As a user, I want to be able to capture skin images with an external tool.	Necessary
4	High-level requirement	The software should be up and running within 2 minutes.	Necessary

Other functional requirements

1. Image capture
 - a. Connect to the external tool.
 - b. Take a picture through the software's user interface.
 - c. Store the image locally for further processing.
2. Image processing
 - a. Preprocess images.
 - b. Utilize the Machine Learning (ML)-pipeline to analyze the images.
 - c. The ML-pipeline provides classification results.
3. Results
 - a. Display the results of the image analysis.
 - b. Provide the option to save the results.

Non-functional requirements

Quality Feature	Description
App's loading time	The app should load within 60 seconds.
Reaction time	When navigating, a reaction should not take longer than 10 seconds.
Design	The design should be modern, simple and intuitive.
Maintainability	The software has to be maintainable to ease further development and use.
Specificity	True negative rate measures the proportion of negatives (harmless skin lesions) that is correctly classified.
Sensitivity	True positive rate measures the proportion of positives (harmful skin lesions) that is correctly classified.
Information security	The patient's information must be secure and protected.
Laws	The app development is restricted by the Medical Device Regulation (MDR) since it is a medical device software. The project is focused on the software life cycle. Consequently, other norms and laws for MDSW are not considered. The project is realized in academia considering the academic capabilities.
Costs	This project is a university project within the scope of a bachelor's thesis. The software and programs must be free since no project budget is available.
Time	The whole thesis takes 3 months. The app development may not take longer than a month, excluding researching.

Constraints

1. Use a ML framework that is compatible with the ML-pipeline.
2. Is compatible with Windows. (see *Runtime environment* below)
3. Is developed by mid February 2024.
4. Complies with regulations for medical device software.

5. User data and images are protected.
6. Is only allowed for use by medical professionals.
7. Is available in English only.
8. Allows for possibilities to integrate other API's, for example User Management Systems.

User interface requirements (UI Specification; after [UI Planning](#))

Functional and non-functional design requirements

Derived from user needs, preferences, capabilities (according to [Use Specification](#)):

- The display shall be visible at a distance of 50 cm to the user.
- Text shall be at least 9 point or larger to ensure legibility among individuals with less than normal visual acuity (e.g. users who are farsighted and might not be wearing their leading glasses).

UI requirements associated with the implementation of risk control measures

Similar to risk control in the risk management process of ISO 14971, the usability norm requires to implement at least one of the risk control measures, which are listed according to their priority in the table below [\[7\]](#), table 10]:

Risk Control Measure Type	Description	Examples
Inherent safety by design	User interface is designed in a way that prevents use errors	<ul style="list-style-type: none"> • System does not allow incorrect input or selection • Improve detectability or readability of controls • Remove features that can be mistakenly selected
Protective measure	Built-in protections against use errors as part of the user interface	<ul style="list-style-type: none"> • Dialogue that requires the user to actively confirm a critical action in order to proceed
Information for safety	Warnings or instructions on correct use that inform the user on potential risks	<ul style="list-style-type: none"> • Training sessions • Online user manual • Highlight problematic inputs / results without requiring any confirmation

If possible, *Inherent safety by design* should be the preferred measure because a redesign of the user interface will most likely reduce or remove the risk [\[7\]](#). However, this might only sometimes be possible. Therefore, the second option is protective measures. In our case for software, there are no protective

measures in the actual sense, such as protective insulation. Instead, *confirmation prompts* can be implemented for critical actions that the user must actively confirm to proceed.

UI Requirements for accompanying documentation and training

Provide the user with a small use manual on the basic operations of the software, only if UI use is not obvious.

UI Requirements for design principles, heuristics, and style guides

Not required. Just make it pretty. 😊

Technical interface

1. General human interface devices.
 - a. Computer with USB-port.
 - b. Display.
 - c. Mouse.
 - d. Keyboard.
 - e. USB-cable for external dermatoscope.

Runtime environment

- Operating system: Windows 10.
- Processor (CPU): Intel Core i7-8250U (8th Gen) or equivalent mid-range processor.
- RAM: 8 GB DDR4.
- Disk storage: 1 TB SSD, no immensely large data processing involved.
- Python version: Python 3.8.2, as the software is developed and tested with this version.
- Additional software: None required.
- Additional libraries and packages: Find [here](#).

A.15. System and Software Design

Software for Skin Cancer Detection – Software for Skin Cancer Detection

System and Software Design

Epics and their issues

1: Welcome page

- App layout
- Welcome text
- Start process button

2: Data page

- User info class
- User data entries
- Import data button
- Proceed button
- Back button

3: History page

- View image and results history
- Add new image button
- Back button

4: Image Preview page

- Capture image button
- Results chart
- Save results and image button
- Restart button

5: App icon

- App icon

Acceptance criteria

1. All unittests pass.
2. Test coverage of at least 50%.
3. Risk score calculations are reproducible.
4. Stakeholder satisfied.

A.16. Software Architecture Description

Software for Skin Cancer Detection – Software for Skin Cancer Detection

Software Architecture Description

General Design Decisions

- The main two colors of the software are white and violet [#CB89FF].
- Information texts have a font size of 14 or 16.
- Headlines have a font size of 28 (subheadings 26, 20, 18, and so on).
- The same font type is used throughout.

App Icon

- The app icon is a violet square with rounded edges.
- There is a cartoonified magnifying glass in the middle.
- The magnifying glass zooms in on a mole on a skin.



Fig. 1: App Icon. Created using figma.

Screen design

- A rectangle window, white background, violet buttons and white background. Black or white font.
- Prototypes can be found in [UI Design and Implementation](#) step and in [Mockup](#).

Software maintenance

- Post-delivery maintenance.
- E-mail address of developer is provided for reports of observed errors, feedback.

Detailed window design

- Rectangle window.
- “Software for Skin Cancer Detection” is the name of the window.
- Option to minimize, reshape or close the window at the top right.

Detailed design of the welcome screen

- Headline reading “Welcome!”
- Welcome text.
- Instruction text after welcome text.
- “Start”-button at the bottom right.

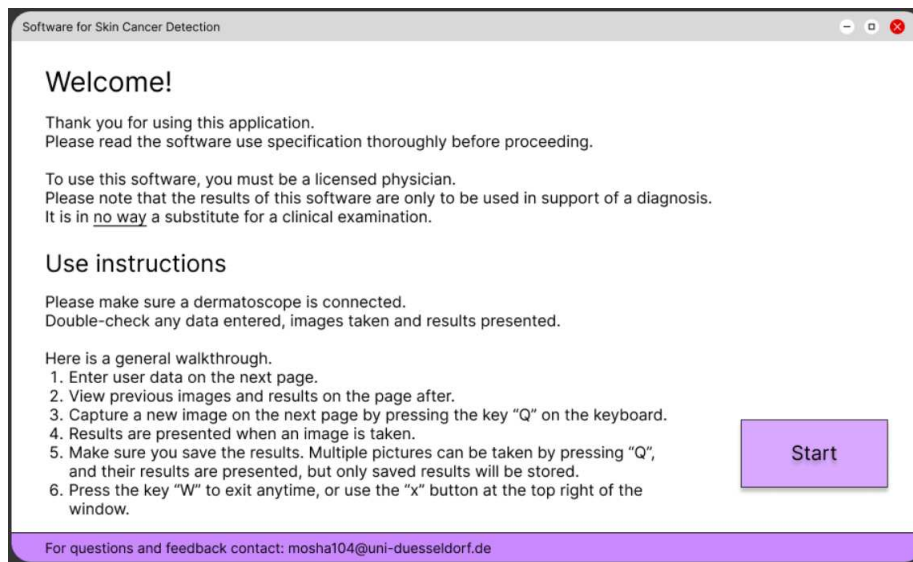


Fig 2: Welcome Screen Mockup. Created using figma.

Detailed design of the data input screen

- Headline reading “User information”.
- Several input-fields, including:
 - First name, Last name, Birth date, Home address, E-mail address, Phone number, Additional notes
- “Import data”-button at the bottom in the middle.
- “Proceed”-button at the bottom right.
- “Back”-button at the bottom left.

The mockup shows a web application window titled "Software for Skin Cancer Detection". The main heading is "User information" with a subtext "Please input or import user info below.". The form is divided into two columns. The left column contains input fields for "First name", "Last name", "Home address", "E-mail address", and "Phone number". The right column contains a "Birthdate" field with a date picker (DD/MM/YYYY) and a large text area for "Additional notes". At the bottom, there are three buttons: "Back", "Import data", and "Proceed". A footer bar contains the contact information: "For questions and feedback contact: moha104@uni-duesseldorf.de".

Fig 3: Data Input Screen Mockup. Created using figma.

Detailed design of the history screen

- Headline reading “User history”.
- A list of previous images & results, including:
 - Creation date, Image name, Physician entry, Results.
- “New image”-button at the bottom right.
- “Back”-button at the bottom left.

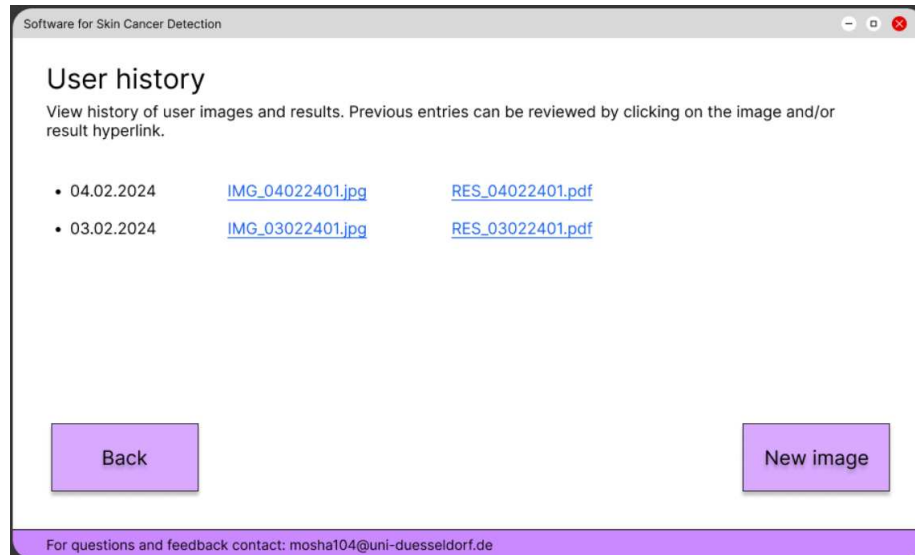


Fig. 4: History Screen Mockup. Created using figma.

Detailed design of the image preview screen

- Headline reading "Capture new photo".
- A square-shaped live preview of camera's view on the left side of the screen.
- A results chart with calculated risk scores on the right side of the screen.
- "Save"-button at the bottom right.
- "Restart"-button.

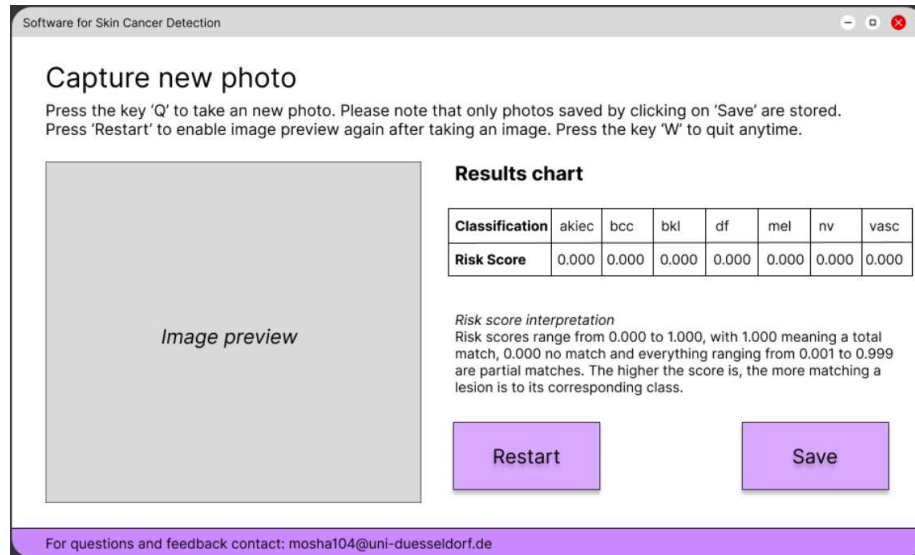


Fig. 5: Image Preview Screen Mockup. Created using figma.

Detailed design of result chart

- A table with seven columns and two rows, excluding heading-column.
- The first row heading reads "Classification".
- The second row heading reads "Risk score".
- Risk score explanation-text under the table.

Detailed design of the project structure

View [Mockup and Class Diagram](#).

A.17. Configuration and Change Management

Software for Skin Cancer Detection – Software for Skin Cancer Detection

Configuration and Change Management

Configuration Management

- Assign unique names or version numbers to different versions of the software.
- Keep a log of changes made to each version. (GitLab issues)

Change Management

1. Include details like the nature of the change and why the change is needed.
2. Present change requests to the stakeholders for approval.
3. Consider factors like impact, resources, and project goals.
4. Make approved changes to the software.
5. Document the changes and test them.
6. Update the software documentation.
7. Version control (Git) to track changes and different software versions.

A.18. Iterative Design Cycle 3 and 4

Software for Skin Cancer Detection – Software for Skin Cancer Detection

3: UI Design and Implementation

Choice of UI elements

- Buttons for navigating through the app or performing actions like loading user info or taking a photo.
- Text fields for inputting text, e.g. user first name.
- Drop-down menu for inputting correct birthdate format.
- Hypertext for accessing a previously saved user image.
- Image preview window for previewing an image before capturing.

Placement of GUI elements

- Buttons are placed so that a flow system is achieved:
 - Buttons on the right are meant for moving *forward*.
 - Buttons placed on the left are for going *back*.
 - Middle buttons perform *actions*.
- Buttons are placed at the bottom of the screen consistently.
- Text fields are placed directly to the right or below their respective labels. A text field may also have its label pre-written inside.
- Texts belonging to another GUI element are placed immediately under or next to it, following the proximity principle in graphic design

Visual configuration

- Color scheme: White, black, and violet.
- Readable and simplistic font, e.g. Arial.
- Size, color and styling are defined in [Software Architecture](#) under *General Design Decisions*.

Interaction design

- Minimalist and simple design for user interactions.
- Clear navigation path sufficing the flow system mentioned above in the *Placement of GUI Elements*.
- Loading circle to acknowledge the user interaction if more than three seconds waiting time.

Design prototypes

Given the scope and resources of the project, a low-fidelity prototype technique is chosen for prototyping: Wireframing.

A high-fidelity prototype is created as a [Mockup in the SLC guideline](#).

www.wireframe.cc is the chosen platform for creating wireframes.

Version 1

Wireframe for welcome screen

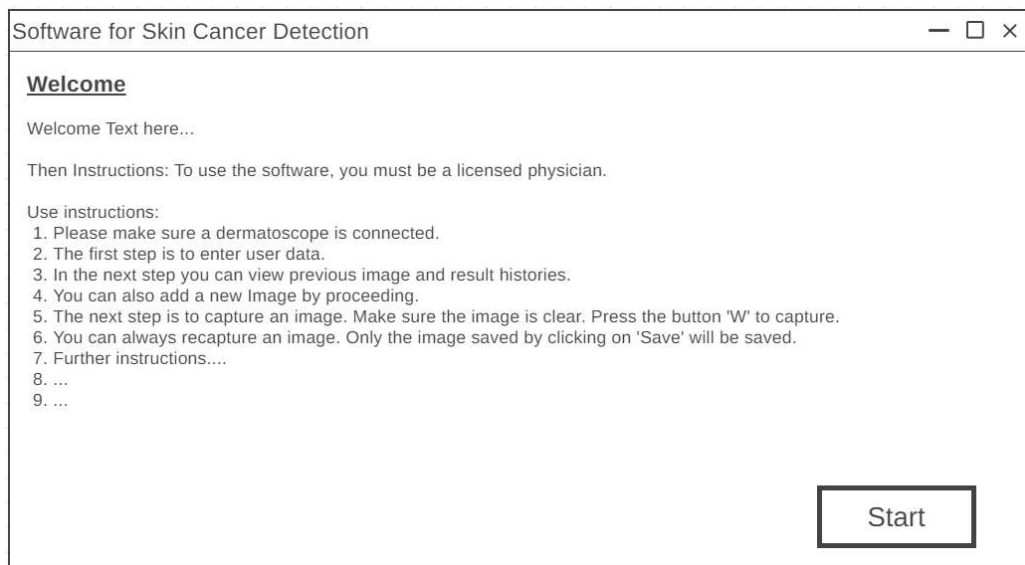
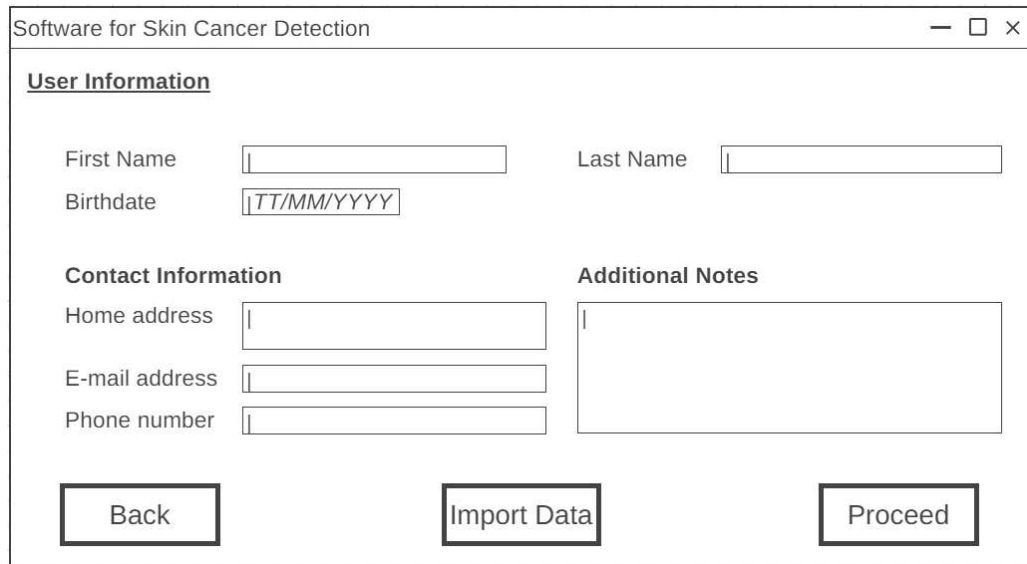


Fig. 1: Wireframe welcome screen.

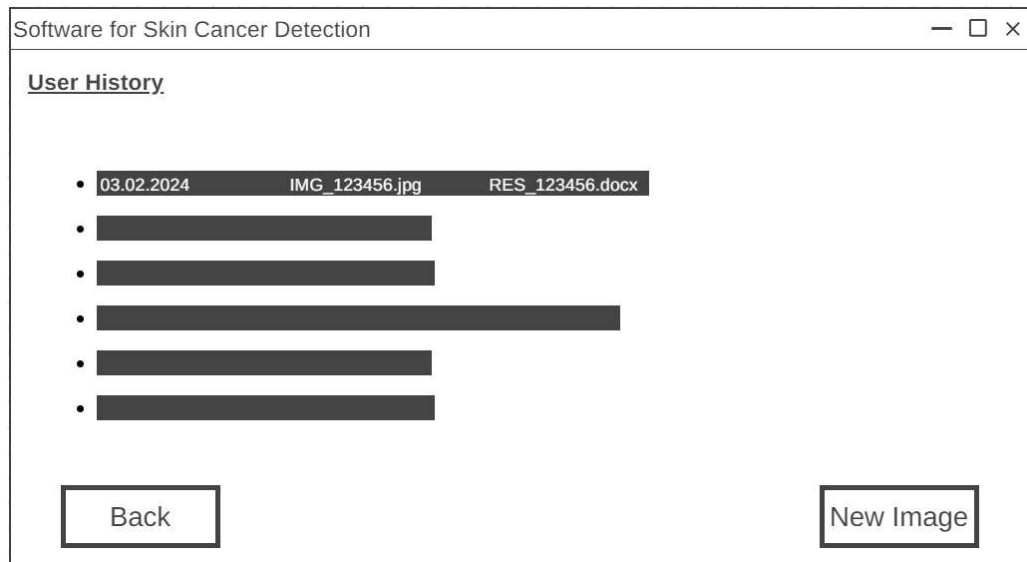
Wireframe for data input screen



The wireframe shows a window titled "Software for Skin Cancer Detection" with standard window controls. Below the title bar is a section header "User Information". It contains three input fields: "First Name", "Last Name", and "Birthdate" (with a date mask "TT/MM/YYYY"). Below this is a "Contact Information" section with three input fields: "Home address", "E-mail address", and "Phone number". To the right of these is an "Additional Notes" section with a larger text area. At the bottom, there are three buttons: "Back", "Import Data", and "Proceed".

Fig. 2: Wireframe data input screen.

Wireframe for history screen



The wireframe shows a window titled "Software for Skin Cancer Detection" with standard window controls. Below the title bar is a section header "User History". It contains a list of items, each represented by a bullet point followed by a dark rectangular bar. The first item is labeled with "03.02.2024", "IMG_123456.jpg", and "RES_123456.docx". Below the list are two buttons: "Back" and "New Image".

Fig. 3: Wireframe history screen.

Wireframe for image preview screen

Software for Skin Cancer Detection

Capture new Photo

Skin Lesion Type	Calculated Risk Score
"Class 0 Lesion Name"	0.000
"Class 1 Lesion Name"	0.001
"Class 2 Lesion Name"	0.921
...	...
...	...
...	...
...	...

Risk score interpretation (draft):
Risk scores range from 0.000 to 1.000, with 1.000 meaning a total match, 0.000 no match and everything ranging from 0.001 to 0.999 are partial matches. The higher the score, the more matching a lesion is to its corresponding class.

Restart

Save

Fig. 4: Wireframe image preview screen.

Final - Version 2

Screenshot of welcome screen

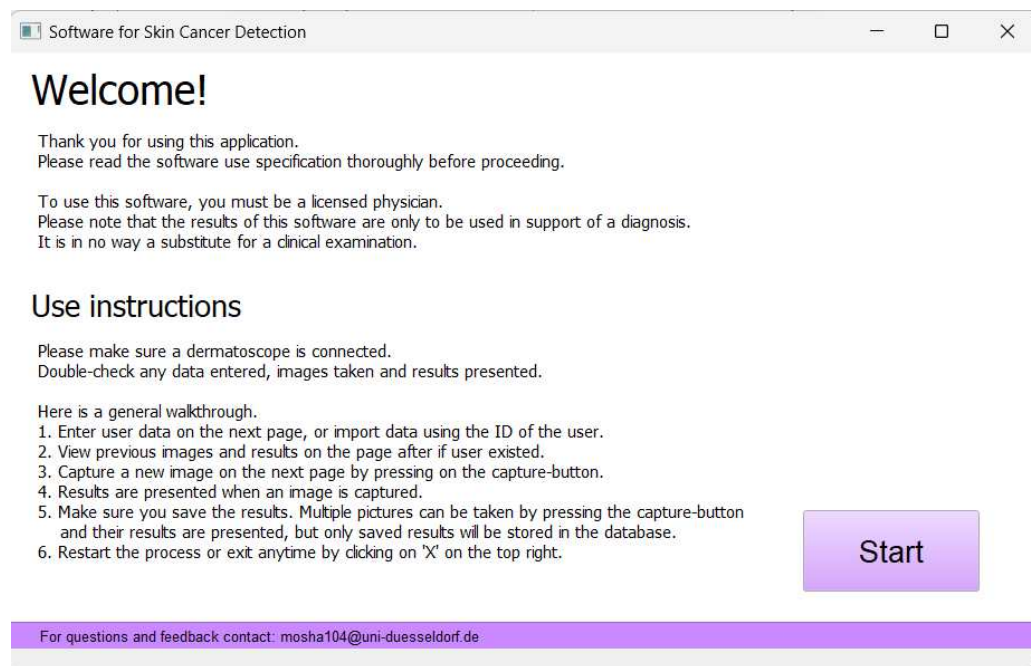


Fig. 5: Welcome screen final version.

Screenshot of data input screen

The screenshot shows a web application window titled "Software for Skin Cancer Detection". The main heading is "User information". Below it, a paragraph states: "Please input new user info below, or import an existing user. Please double check entries before proceeding. Contact information and additional notes will be updated if new info is provided or old info is removed." The form is divided into two main sections: "User information" and "Contact information".

User information

First Name* Birthdate*

Last Name*

Contact information

Home address

E-mail address

Phone number

Additional notes

At the bottom of the form, there are three purple buttons: "Back", "Import data", and "Proceed".

For questions and feedback contact: mosha104@uni-duesseldorf.de

Fig. 6: data input screen final version.

Screenshot of history screen

Software for Skin Cancer Detection

User history

View history of user images and results. Previous images can be viewed by clicking on the image name in the corresponding cell.
If this is empty, there are no previous entries for the user.

	Date	Image name	Physician entry	akiec	bcc	bkl	df	mel	nv	vasc
1	11-02-2024	IMG_1	n/a	0.0012	0.0002	0.005	0.0086	0.0005	0.9594	0.025
2	11-02-2024	IMG_7	bcc	0.0011	0.0002	0.0049	0.0082	0.0004	0.9623	0.0229
3	11-02-2024	IMG_9	mel	0.0012	0.0002	0.0049	0.0085	0.0004	0.9607	0.0241
4	11-02-2024	IMG_11	vasc	0.0006	0.0002	0.0158	0.0024	0.0012	0.9709	0.0088
5	11-02-2024	IMG_13	vasc	0.0012	0.0002	0.0063	0.0095	0.0005	0.9562	0.0259
6	14-02-2024	IMG_14	n/a	0	0	0	0	0	1	0

Back **New image**

For questions and feedback contact: mosha104@uni-duesseldorf.de

Fig. 7: History screen final version.

Screenshot of image preview screen

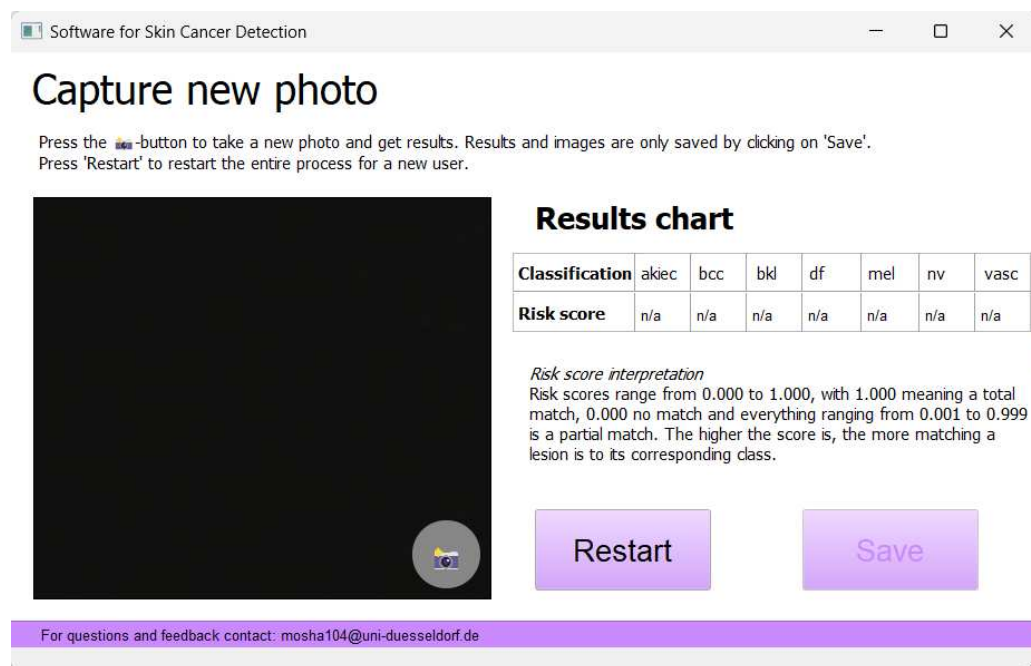


Fig. 8: Image preview screen final version.

4: Formative Evaluation

First formative evaluation, 06.02.2024

Objective	Initial evaluation
Evaluation Method	Systematic inspection of wireframe
Evaluation Focus	Overview, space
Time plan	One day

After evaluating the initial [wireframe prototype](#), small changes have been made.

1. Swapping of “Last name” and “Birthdate” in data input screen. *Reason:* Similar labels must be located closer to each other than other labels.
2. Change of “Birthdate” input format from “TT/MM/YYYY” to “DD/MM/YYYY”. *Reason:* Mix of German and English language.
3. Adding small info text on each screen. *Reason:* Better user experience and error minimization.
4. Transforming table in image preview screen from seven rows to two rows and from two columns to seven columns. *Reason:* Space issues.

Second formative evaluation, after programming UI, 13.02.2024

Objective	Programming adjustment evaluation
Evaluation Method	Usability tests
Evaluation Focus	UI appearance and adjustments
Time plan	Three days

After programming the interface and its functionality, a couple changes have been made:

1. Adding “*” to required inputs in data input screen.
2. “Import data”-button now asks for a userId to load the user in an external window.
3. User history screen is now a table instead of a list for overview and functionality purposes.

4. New “Capture”-button on image preview screen instead of using the key “Q”.
5. “Save”-button on image preview screen is disabled until a picture is captured to minimize software error risks.

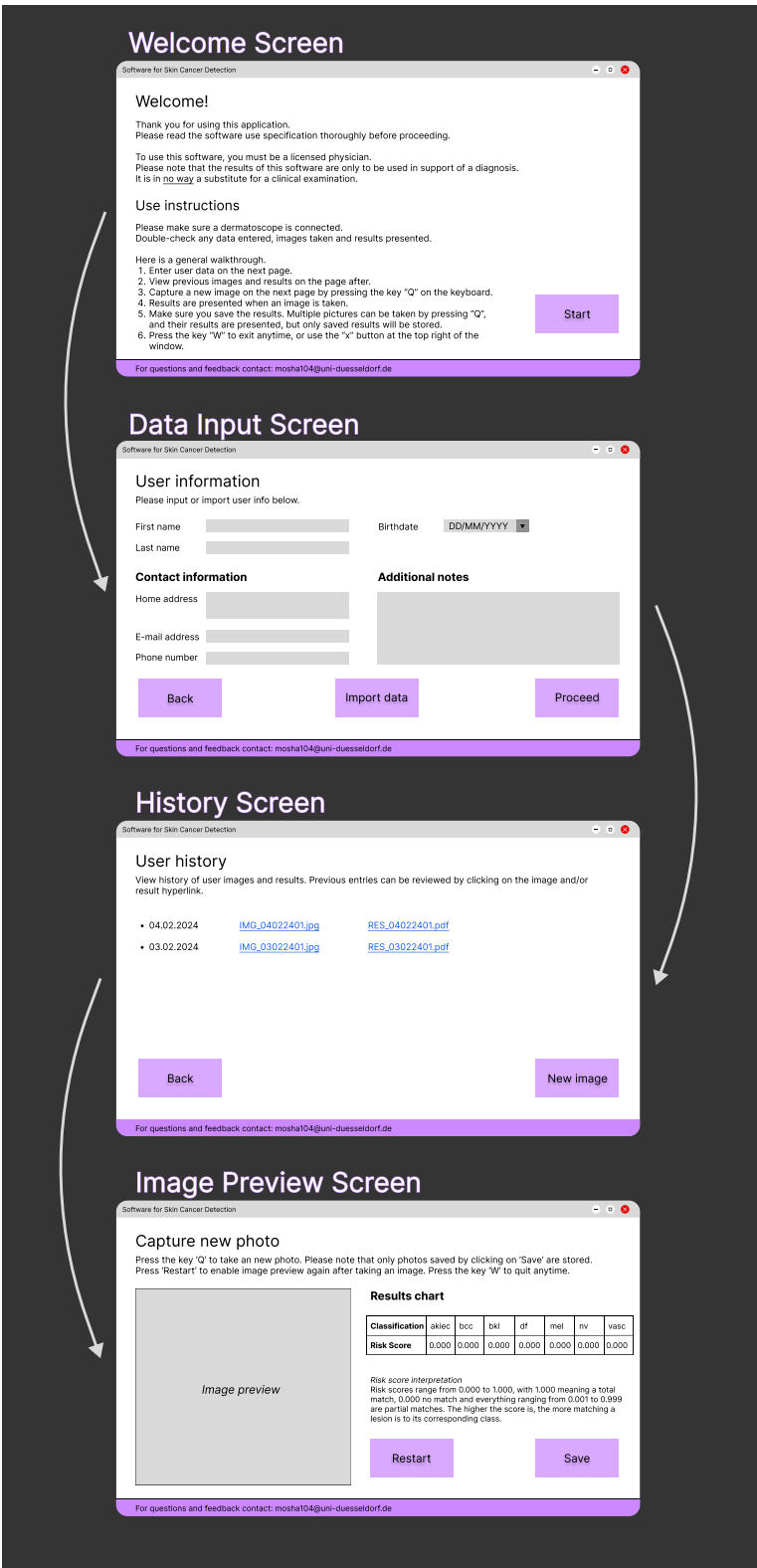
Third formative evaluation, 16.02.2024

Objective	Programming adjustment evaluation II
Evaluation Method	Usability tests, user experience tester
Evaluation Focus	Final UI appearance and adjustments, fulfillment of high-level UI requirements
Time plan	Two days

A couple changes have been made, fulfilling the wishes of the stakeholder:

1. “Import data”-button now asks for the first and last name of the user to load the user in an external window, instead of asking for the userId. This makes it easier to find a user by name if the ID is unknown.
2. The table in history screen now has another column, physician entry, in which the physician can choose which diagnosis has been made via a drop-down menu. This is a prerequisite for the planned use of the stakeholder for data collection purposes.

A.19. Mockup Process



A.20. Class Diagram



A.21. Use-related Risk Analysis

Software for Skin Cancer Detection – Software for Skin Cancer Detection

Use-related Risk Analysis

Find here a risk sheet of a collection of potential use errors, hazards and situations related to usability.

Identification of safety-related UI characteristics

List of safety-related UI characteristics and potential use errors

ID	Safety-related UI characteristics	Use error	Root cause
U1	Upload image	User uploads an image that does not follow the requirements.	User might have overseen the information on the requirements.
U2	Add user info	User loads or adds wrong info as user data.	User input wrong userId.
U3	Does not save Results	User forgets to save results.	User might not have read the informative text.

Analysis of all possible consequences triggered by human errors

ID	Use-related hazard	Hazardous situation
UH1	False prediction of the ML-pipeline results in a wrong medical decision on a patient	Patient receives wrong diagnosis / treatment
UH2	Wrong and/or misleading insights gained from the interaction, results in a wrong medical decision on a patient	Patient receives wrong diagnosis / treatment
UH3	Manipulation of other systems by faulty information	Potential wrong diagnosis because of incorrect data
UH4	Violation of data protection and privacy	Information and privacy data abuse

Usability Engineering Process Guideline – 60

Identification of hazard-related use scenarios

View [Risk Management hazards](#).

Definition of harm & severity levels (see [Risk Policy](#))

Severity level of harm	Description
Negligible	Results in inconvenience or temporary discomfort, no injury
Moderate	Results in injury or impairment requiring or not professional medical care
Significant	Results in permanent impairment or life-threatening injury

Risk control measures

ID	Measure description	Risk control type
R1a	“Proceed”-button is clicked on empty first name user data input. Error message explains that the input cannot be empty and is invalid.	Inherent safety by design
R1b	“Proceed”-button is clicked on empty last name user data input. Error message explains that the input cannot be empty and is invalid.	Inherent safety by design
R2	“Import data”-button is clicked and user submits invalid input or non-existent user. Error message explains that the input is invalid.	Inherent safety by design
R3	“Save”-button cannot be clicked and is disabled unless an image was captured.	Inherent safety by design

Find a final list of hazard-related use scenarios in the [Summative Evaluation](#).

A.22. Summative Evaluation

Software for Skin Cancer Detection – Software for Skin Cancer Detection

Summative Evaluation

Following the [Summative Evaluation plan](#), on the final implementation of the UI, a summative evaluation is performed. Tasks associated with hazard related use scenarios can be conducted successfully and safely on the UI by the intended users in the use environment:

Objective	Check use scenario
Evaluation method	Usability test in a simulated use environment
Acceptance criteria	<ul style="list-style-type: none">• No use error occurred that leads to harm or a specific severity level of harm.• No new hazards, hazardous situations, hazard-related use scenarios were identified.

Every hazard-related use scenario and evaluation

Hazard-related use scenario	Evaluation
U1: User uploads image that does not follow requirements	No risk. Images are only saved by clicking on the respective “Save”-button. It imposes no risk if an image is taken incorrectly, as images can always be recalled in User history and be double checked.
U2: User loads or adds wrong info as user data	One can see which user has been loaded before proceeding. Risk is acceptable.
U3: User does not save results	A new image with new results can be taken again in a matter of minutes. It imposes no risk, only a slight inconvenience.

Usability test protocol

Each hazard-related use scenario is translated into one test scenario. The tests conducted are structured in a way that allows the user to perform a task without getting any hints. For the tests, two users with an academic background were invited to perform the tasks. Both performed the tasks almost identically.

Test ID	Test	Hazard-related use scenario	Evaluation
T1	View entry history of user with the name "Tester, TestUser".	U2	<p>The user's task was to show us the history of the user with the name "Tester, TestUser". He was not provided with any additional material except the running software itself.</p> <p>The user went on to read the introductory text and use instructions on the welcome screen. He then clicked on the "Start"-button and landed on the data input screen. There, he read the informative text and clicked on the "Import data"-button; shortly after, the dialog input field opened, requesting a full name. After inputting the first letters of the user name: "teste", the software autorecommended the name "Tester, TestUser" and the user confirmed by pressing Enter. He clicked on the "Ok"-button, leading to the closing of the dialog input window and the importing of the user data. After double checking the first- and last name imports in the line edit fields, the user clicked on the "Proceed"-button and was able to view the entry history of the user.</p> <p>The second user was provided with the same material and performed the task identically with one small difference: Instead of inputting the first letters of the user name into the dialog input window, the user clicked on the drop-down menu and simply chose the user "Tester, TestUser".</p> <p>No risks, errors or difficulties were observed.</p>
T2	Following T1, upload a new image for the user and save it.	U1, U3	<p>The user was provided with a physical image of a skin lesion. He then was instructed to add this image as an entry for the user he imported in T1.</p> <p>The user clicked on the "New image"-button. He landed on the image preview screen and paused to read and comprehend the new page for a good minute before grabbing the image and clicking on the "Capture"-button. The user noticed that results changed from n/a to numerical values. He proceeded to click on the now activated "Save"-button, deactivating it again. This reassured the user that the image was saved.</p> <p>No risks, errors or difficulties were observed. However, one of the users added a valid point to their feedback, encouraging the developing team to display an "Image saved successfully."-text after saving an image for confirming.</p>

Following this summative evaluation, it is concluded that the software UI is not in urgent need of any additional editing or changing as no use errors, *unacceptable risks* or difficulties have been reported. Both acceptance criteria were met.

This results in not transferring this summative evaluation into a formative one and therefore finishing the UE guideline's UEIDC successfully.

A.23. Risk Monitoring

Software for Skin Cancer Detection – Software for Skin Cancer Detection

Risk Monitoring

Updated risk table, 19.02.2024

Risk ID	Component	Error (Cause)	Hazard	Damage	Implemented?
R1	Database	Database not reachable or takes too long to answer	H1	No (or delayed) treatment	No
R2	Application	Incomplete form submission (missing inputs)	H1	No (or delayed) treatment	Yes
R3	Database Security	Malicious SQL injection	H1, H2, H3, H4	No treatment, false treatment, system manipulation or user data violation	Yes
R4	Database Security	Unauthorized access to stored patient diagnoses	H4	Privacy violation	No
R5	Database Security	Malicious user floods system with fake data	H3	System damage	No/Obsolete
R6	Application	User provides wrong name	H1, H2	No or false treatment	Yes
R7	Application	User uploads garbage data (e.g. unrelated images)	H1, H2	No or false treatment	Partially
R8	Communication	User provides wrong contact info	H1, H2, H4	No treatment, false treatment or privacy violation	No/Obsolete

Updated risk matrix, 19.02.2024

	Negligible	Moderate	Significant
High	0	0	0
Medium	0	0	0
Low	1	0	2

Residual risks

All unacceptable risks have been mitigated and acceptable risks can still be mitigated if they become unacceptable in the future due to maintainability. The project hereby is ready for release.

A.24. README.txt

Documentation-based Machine Learning Software for Skin Cancer Detection

Name

Software for Skin Cancer Detection - Part of a Bachelor's thesis

Description

The general purpose of this app is to allow users to capture images of skin lesions and get risk scores for that lesion.

The app works like this:

- 1) Enter user data or import existing user by id. Required: First and last name, birthdate (identifiers).
- 2) Add voluntary Contact information or Additional information. These inputs can be updated for an existing user if user is imported and changes are simply made before clicking on the Proceed-button.
- 3) After proceeding: For existing users, a history of previous entries is displayed. These include:
Date of entry, IMG name, and risk scores for each classification.
For new users, go to 6).
- 4) By clicking on the cell containing the IMG name, the image is opened as a preview.
- 5) Click on the New Image-Button to start Image preview and take a new picture + entry.
- 6) The camera opens and you are able to capture an image of a lesion by clicking on the Capture-button.
- 7) Results are displayed then, and images and results can then be saved by clicking on Save.
Multiple pictures can be taken, only Images + Results are saved if the Save-button is clicked.
- 8) After finishing, one may either just close the app or click Restart in order to add entries for other users.

Installation

In order to run this app, please refer to the Wiki for technical requirements.

This app runs on PyCharm. Just start the Software.py application and import all needed packages.

Before starting, make sure to specify the correct filepath for images and results to be saved, or else the tests won't run.

Change the fpath-variable in both TestSoftware.py and Software.py to your desired path. Also, set all (2 code lines) cv

camera instances to 1 instead of 0, if the connected dermatoscope is your second camera. Comments clarify.

In the database, do not remove the first user in users-table (TestUser Tester). He is crucial for test purposes.

Also do not remove the first Image Entry (IMG_1) neither here nor in local folders. It is also crucial for testing.

The rest is voluntary.

If you choose to delete imagesAndResults entries, make sure to also delete the corresponding IMG and RES files in your local folder!

Do this in the root directory (software_for_skin_cancer_detection): 'pip install .' to install the package.

Support

Contact mosh104@hhu.de for questions, feedback, support and potential access to the Wiki.

Authors and acknowledgment

Thank you, Jan Ruhland and Roman Martin, for your supervision and helpful assistance both in developing this software and in my thesis work!

License

Most imports and packages are open source. Licensed libraries/modules, as the PyQt5, are licensed under GPL/LGPL, which

both allow the use for academia & private programming. Details in Wiki under Licenses.

Project status

Project is finished, some tests still need implementing & finishing aesthetics must be implemented, however, the most important functionalities, tests and design work. Ready for release.

A.25. Software Problem-solving Process

Software for Skin Cancer Detection – Software for Skin Cancer Detection

Software problem-solving process

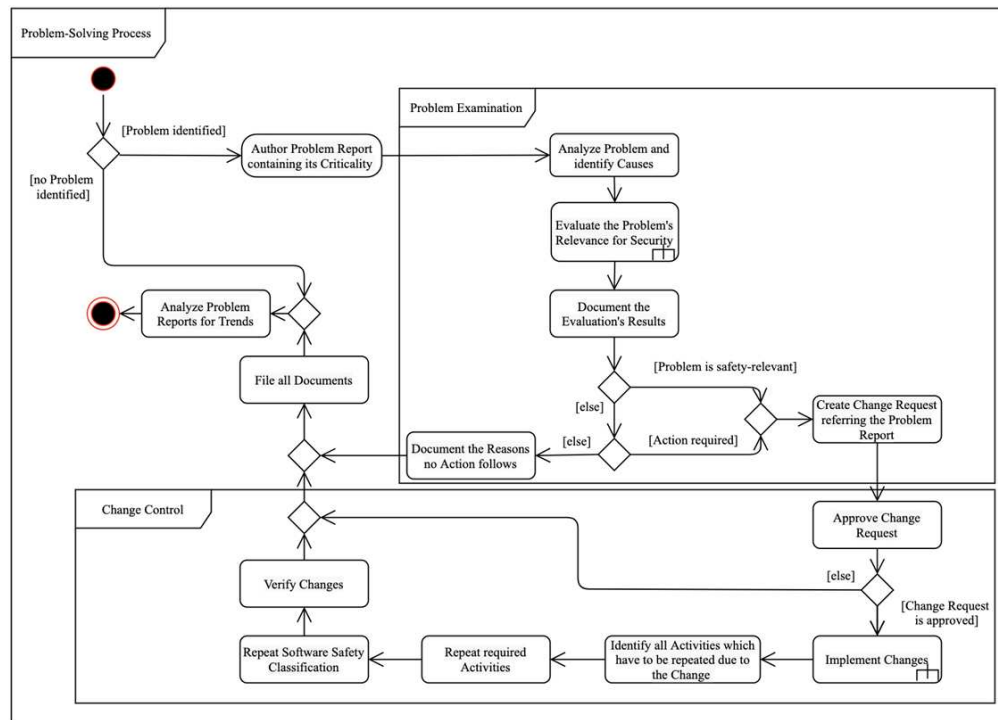


Fig. 1: Problem-solving process from [5, Figure 24]. Starting at the top left, if a problem is identified, follow the steps and perform the actions depicted. It must lead to the circle with a red line around.

A.26. Confluence Bibliography

Software for Skin Cancer Detection – Software for Skin Cancer Detection

Bibliography

1. Hauschild, A.-C. • Eick, L. • Wienbeck, J. • Heider, D. (2021): **Fostering reproducibility, reusability, and technology transfer in health informatics**. *iScience*. <https://doi.org/10.1016/j.isci.2021.102803>
2. Hauschild, A.-C. • Martin, R. • Holst, S. C. • Wienbeck, J. • Heider, D. (2022): **Guideline for software life cycle in health informatics**. *iScience*. <https://doi.org/10.1016/j.isci.2022.105534>
3. Martin, R. • Hauschild, A.-C. • Gottschalk, R. • Clemens, S. • Wienbeck, J. • Heider, D. (2024; in progress): **Guideline for Risk Management in Software Health and Medical Applications**. *[internal unreleased document; in progress]*.
4. Heider, D. • Hauschild, A.-C. • Wienbeck, J. • Martin, R. • Clemens, S. • Klemt, V. (2023): **Privacy preserving federated machine learning and blockchaining for reduced cyber risks in a world of distributed healthcare**. *FeatureCloud*. *[internal unreleased document; in progress]*.
5. Holst, S. C. (2021): **Software Life Cycle Guideline for Software as a Medical Device - Implementing a Prototype App**. Master's thesis at Philipps University Marburg.
6. Gottschalk, R. (2021): **Risk Management Guideline for Software as a Medical Device and Example Implementation for a Federated Learning Application**. Master's thesis at Philipps University Marburg.
7. Klemt, V. (2021): **Usability Engineering Guideline for Software as a Medical Device – Implementing an Interactive xAI Platform supporting Medical Decision-Making**. Master's thesis at Philipps University Marburg.